Advanced Universal Video Analysis System

Project Overview

This system uses AI to deeply analyze any video content by combining:

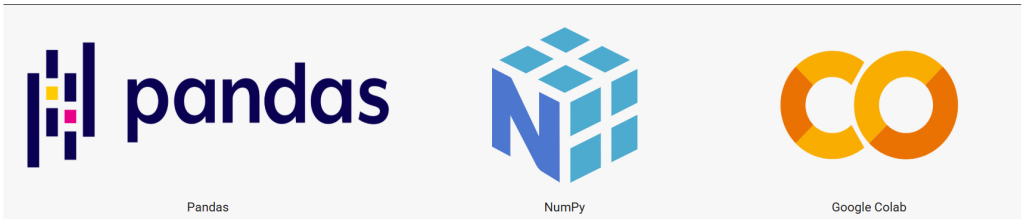
- **Object detection** with YOLO v8
- **Scene understanding** with Google's Gemini Vision API
- **Intelligent conversation** for natural language video exploration

Key Technologies

YOLOv8 Google Gemini OpenCV

YOLOv8 Google Gemini OpenCV

Pandas NumPy Google Colab



Pandas NumPy Google Colab

Applications:

- Content analysis for media companies
- Automated video summarization
- Specialized domain analysis (wildlife, sports, entertainment, etc.)
- Accessibility features for visually impaired users(when voice is added in further iteration of development)

[2] `!pip install ultralytics opencv-python --quiet`

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 56.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 107.9 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 61.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 57.8 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 14.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.9 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 59.7 MB/s eta 0:00:00
```

[4]
```
uploaded = files.upload()

video_path = list(uploaded.keys())[0]  # e.g., 'traffic.mp4'
print(f"Video uploaded: {video_path}")
```

```
Choose files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Cheetah cub learns how to hunt and kill a scrub hare.mp4 to Cheetah cub learns how to hunt and kill a scrub hare.mp4
Video uploaded: Cheetah cub learns how to hunt and kill a scrub hare.mp4
```

[5]
```
import cv2
import os
import json
import numpy as np
from collections import defaultdict
import google.generativeai as genai
from ultralytics import YOLO
from PIL import Image
import base64
import io
from google.colab import files
import shutil
```

System Configuration

This cell sets up:

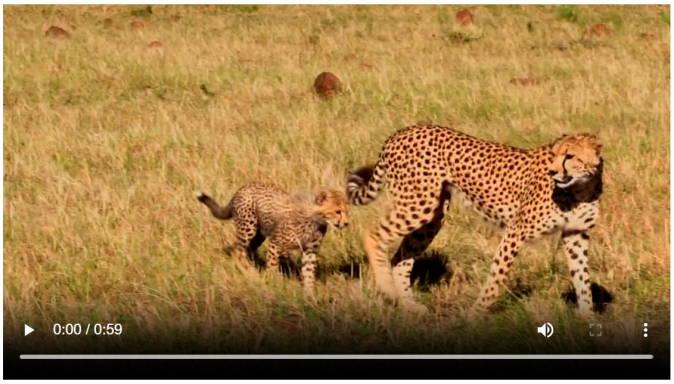
1. **API Keys**: Configure access to Gemini Vision API

2. **Analysis Parameters**:
    - **Interval**: How frequently to analyze frames (balances detail vs processing time)
    - **Confidence Threshold**: Minimum detection confidence for YOLO
    - **Scene Change Sensitivity**: How dramatically frames must differ to detect a scene change

IMPORTANT: Replace the placeholder API key with your own Gemini API key as we used our key which is coming at some cost.

```
[19] import os
     GEMINI_API_KEY = os.environ.get("GEMINI_API_KEY")

     genai.configure(api_key=GEMINI_API_KEY)

     ANALYSIS_INTERVAL = 5.0
     CONFIDENCE_THRESHOLD = 0.5
     SCENE_CHANGE_THRESHOLD = 0.3
```

```
[20]
     from IPython.display import HTML
     from base64 import b64encode

     def display_video(video_path):
         """Display the video in the notebook"""
         video_file = open(video_path, "rb").read()
         encoded = b64encode(video_file).decode('ascii')

         return HTML(f"""
         <div style="display: flex; justify-content: center; margin: 20px 0;">
             <video width="640" height="360" controls>
                 <source src="data:video/mp4;base64,{encoded}" type="video/mp4">
                 Your browser does not support the video tag.
             </video>
         </div>
         <div style="text-align: center; font-style: italic; margin-bottom: 30px;">
             Video: {video_path}
         </div>
         """)

     # Display the video
     display_video(video_path)
```



Video: Cheetah cub learns how to hunt and kill a scrub hare.mp4

```
[8]
     cap = cv2.VideoCapture(video_path)
     fps = cap.get(cv2.CAP_PROP_FPS)
     total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
     duration = total_frames / fps
     cap.release()


     print(f" Duration: {duration:.1f} seconds")
     print(f" FPS: {fps:.1f}")
     print(f" Total frames: {total_frames}")
```

```
 Duration: 59.3 seconds
 FPS: 30.0
 Total frames: 1778
```

Universal Video Analysis Engine

This is the core intelligence of our system. The `UniversalVideoAnalyzer` class:

Key Components:

1. **Object Detection**
    - Uses YOLO v8 to identify objects, people, animals, etc.
    - Tracks detections with timestamps and confidence scores
2. **Scene Change Detection**
    - Automatically identifies transitions between scenes
    - Uses computer vision algorithms to detect significant frame differences

3. **Advanced Scene Understanding**
    - Uses Gemini Vision for human-like understanding of:
        - Subjects and their attributes
        - Activities and behaviors
        - Settings and environments
        - Technical elements (cinematography, lighting)
        - Emotional tone and narrative context
4. **Timeline Generation**
    - Creates a chronological summary of key events
    - Timestamps important moments for easy navigation

The analyzer works with any generic video content type

```python
class UniversalVideoAnalyzer:
    def __init__(self, gemini_api_key):
        """Initialize a universal video analyzer for any content type"""
        self.yolo_model = YOLO("yolov8n.pt")

        self.vision_model = genai.GenerativeModel("gemini-2.0-flash-exp")

        self.frame_analyses = []
        self.scene_changes = []
        self.objects_timeline = defaultdict(list)

    def detect_scene_change(self, prev_frame, curr_frame, threshold=0.3):
        """Detect significant scene changes"""
        if prev_frame is None:
            return True

        # Convert to grayscale for comparison
        prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
        curr_gray = cv2.cvtColor(curr_frame, cv2.COLOR_BGR2GRAY)

        diff = cv2.absdiff(prev_gray, curr_gray)
        diff_score = np.mean(diff) / 255.0

        return diff_score > threshold

    def analyze_frame_with_gemini(self, frame, timestamp):
        """Use Gemini Vision to understand any type of scene"""
        try:
            # Convert frame for Gemini
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            pil_image = Image.fromarray(frame_rgb)

            # Universal prompt that works for ANY video content
            prompt = f"""
            Analyze this video frame (timestamp: {timestamp:.1f}s) in detail.

            Provide a comprehensive analysis covering:

            1. SUBJECTS: All key people, animals, or main subjects visible
            2. ACTIVITY: Primary actions, events, or behaviors taking place
            3. SETTING: Location, environment, time of day, overall context
            4. TECHNICAL: Notable cinematography, lighting, composition elements
            5. INTERACTIONS: Relationships between subjects, social dynamics
            6. EMOTIONAL: Mood, atmosphere, emotional content of the scene
            7. NARRATIVE: How this moment fits into a potential story arc
            8. NOTABLE: Any unusual, distinctive or significant details

            Respond in JSON format:
            {{
                "subjects": ["detailed descriptions of key subjects"],
                "activities": ["primary actions happening"],
                "setting": "comprehensive setting description",
                "technical_elements": ["notable visual/production elements"],
                "interactions": ["relationship dynamics observed"],
                "emotional_tone": "mood and emotional qualities",
                "narrative_context": "story context inference",
                "notable_elements": ["unique or significant details"],
                "genre": "apparent content type (documentary, sports, drama, etc.)",
                "summary": "one sentence comprehensive description"
            }}
            """

            response = self.vision_model.generate_content([prompt, pil_image])

            # Parse JSON response
            try:
                analysis = json.loads(response.text.strip().replace('```json', '').replace('```', ''))
            except:
                # Fallback if JSON parsing fails
                analysis = {
                    "subjects": [],
                    "activities": [],
                    "setting": "unknown",
                    "technical_elements": [],
                    "interactions": [],
                    "emotional_tone": "neutral",
                    "narrative_context": "unknown",
                    "notable_elements": [],
                    "genre": "unknown",
                    "summary": response.text[:100] + "..."
                }
```

```
            analysis["timestamp"] = timestamp
            return analysis

        except Exception as e:
            print(f"Gemini analysis failed at {timestamp:.1f}s: {e}")
            return {
                "timestamp": timestamp,
                "subjects": [],
                "activities": [],
                "setting": "unknown",
                "technical_elements": [],
                "interactions": [],
                "emotional_tone": "neutral",
                "narrative_context": "unknown",
                "notable_elements": [],
                "genre": "unknown",
                "summary": "Analysis failed"
            }

    def detect_objects_yolo(self, frame, timestamp):
        """Use YOLO for object detection"""
        results = self.yolo_model.predict(frame, conf=CONFIDENCE_THRESHOLD, verbose=False)

        detected_objects = []
        for r in results:
            if r.boxes is not None:
                for box in r.boxes:
                    cls_id = int(box.cls[0])
                    label = self.yolo_model.names[cls_id]
                    conf = float(box.conf[0])

                    detected_objects.append({
                        "object": label,
                        "confidence": conf,
                        "timestamp": timestamp
                    })

                    # Add to timeline
                    self.objects_timeline[label].append({
                        "time": timestamp,
                        "confidence": conf
                    })

        return detected_objects

    def analyze_video(self, video_path, analysis_interval=5.0):
        """Analyze the entire video"""
        cap = cv2.VideoCapture(video_path)
        fps = cap.get(cv2.CAP_PROP_FPS)
        total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
        duration = total_frames / fps

        print(f" Starting analysis of {duration:.1f}s video...")
        print(f" Will analyze every {analysis_interval}s")

        frame_count = 0
        prev_frame = None
        last_analysis_time = -analysis_interval

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            timestamp = frame_count / fps

            # Check for scene changes every second
            if frame_count % int(fps) == 0:
                if self.detect_scene_change(prev_frame, frame):
                    self.scene_changes.append({
                        "timestamp": timestamp,
                        "frame_number": frame_count
                    })
                    print(f" Scene change detected at {self.format_time(timestamp)}")
                prev_frame = frame.copy()

            # Perform analysis at intervals
            if timestamp - last_analysis_time >= analysis_interval:
                print(f" Analyzing frame at {self.format_time(timestamp)}...")

                # YOLO object detection
                yolo_objects = self.detect_objects_yolo(frame, timestamp)

                # Gemini vision analysis
                gemini_analysis = self.analyze_frame_with_gemini(frame, timestamp)

                # Store combined analysis
                frame_analysis = {
                    "timestamp": timestamp,
                    "frame_number": frame_count,
                    "yolo_objects": yolo_objects,
                    "gemini_analysis": gemini_analysis
                }

                self.frame_analyses.append(frame_analysis)
                last_analysis_time = timestamp

            frame_count += 1

            # Progress indicator
            if frame_count % (total_frames // 10) == 0:
```

```
                progress = (frame_count / total_frames) * 100
                print(f"📈 Progress: {progress:.0f}%")

        cap.release()
        print("✅ Video analysis complete!")
        return self.generate_content_summary()

    def format_time(self, seconds):
        """Format seconds as MM:SS"""
        minutes = int(seconds // 60)
        secs = int(seconds % 60)
        return f"{minutes:02d}:{secs:02d}"

    def generate_content_summary(self):
        """Generate a comprehensive content summary for any video type"""
        if not self.frame_analyses:
            return "No analysis data available"

        # Extract universal content information
        all_subjects = set()
        all_activities = []
        all_interactions = []
        all_settings = []
        all_emotions = []
        all_notable_elements = []
        all_technical_elements = []
        detected_genres = set()

        for analysis in self.frame_analyses:
            # From YOLO
            for obj in analysis["yolo_objects"]:
                # Add all detected objects as subjects
                all_subjects.add(obj["object"])

            # From Gemini Vision analysis
            gemini = analysis["gemini_analysis"]
            all_subjects.update(gemini.get("subjects", []))
            all_activities.extend(gemini.get("activities", []))
            all_interactions.extend(gemini.get("interactions", []))
            all_notable_elements.extend(gemini.get("notable_elements", []))
            all_technical_elements.extend(gemini.get("technical_elements", []))

            if gemini.get("setting"):
                all_settings.append(gemini["setting"])

            if gemini.get("emotional_tone"):
                all_emotions.append(gemini["emotional_tone"])

            if gemini.get("genre"):
                detected_genres.add(gemini["genre"])

        # Create timeline
        timeline = []
        for analysis in self.frame_analyses:
            timestamp = analysis["timestamp"]
            summary = analysis["gemini_analysis"].get("summary", "")
            timeline.append(f"{self.format_time(timestamp)}: {summary}")

        summary = {
            "video_duration": f"{self.format_time(self.frame_analyses[-1]['timestamp'])}",
            "scene_changes": len(self.scene_changes),
            "subjects_detected": list(all_subjects),
            "activities_observed": list(set(all_activities)),
            "interactions": list(set(all_interactions)),
            "settings": list(set(all_settings)),
            "emotional_tones": list(set(all_emotions)),
            "notable_elements": list(set(all_notable_elements)),
            "technical_elements": list(set(all_technical_elements)),
            "detected_genres": list(detected_genres),
            "timeline": timeline,
            "analysis_points": len(self.frame_analyses)
        }

        return summary

print("🔧 Initializing Universal Video Analyzer...")
analyzer = UniversalVideoAnalyzer(GEMINI_API_KEY)
print("Analyzer ready!")
```

```
🔧 Initializing Universal Video Analyzer...
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt to 'yolov8n.pt': 100%|██████████| 6.25M/6.25M [00:00<00:00, 107MB/s]Analyzer ready!
```

Execute Video Analysis

This cell runs the comprehensive analysis pipeline:

1. **Frame Extraction**: Samples frames at regular intervals
2. **Scene Detection**: Identifies significant scene changes
3. **Multi-Modal Analysis**: Combines YOLO object detection with Gemini Vision
4. **Timeline Creation**: Builds a chronological summary of the video
5. **Content Classification**: Automatically detects the genre/content type

Note: Analysis time depends on video length and complexity. A typical 2-minute video might take 5 minutes to analyze.

Output: When complete, you'll see a summary of all detected elements across the entire video.

[10]
```
print("Starting comprehensive video analysis...")
print("This may take a few minutes depending on video length...")

# Run the analysis
analysis_summary = analyzer.analyze_video(video_path, analysis_interval=ANALYSIS_INTERVAL)

print("\n" + "="*50)
print("VIDEO ANALYSIS COMPLETE!")
print("="*50)

# Display summary
print(json.dumps(analysis_summary, indent=2))
```

```
Starting comprehensive video analysis...
This may take a few minutes depending on video length...
 Starting analysis of 59.3s video...
 Will analyze every 5.0s
 Scene change detected at 00:00
 Analyzing frame at 00:00...
 Analyzing frame at 00:05...
📈 Progress: 10%
 Analyzing frame at 00:10...
📈 Progress: 20%
 Analyzing frame at 00:15...
📈 Progress: 30%
 Analyzing frame at 00:20...
📈 Progress: 40%
 Analyzing frame at 00:25...
📈 Progress: 50%
 Analyzing frame at 00:30...
 Analyzing frame at 00:35...
📈 Progress: 60%
 Analyzing frame at 00:40...
📈 Progress: 70%
 Analyzing frame at 00:45...
📈 Progress: 80%
 Analyzing frame at 00:50...
📈 Progress: 90%
 Analyzing frame at 00:55...
 Scene change detected at 00:59
📈 Progress: 100%
 Video analysis complete!

==================================================
VIDEO ANALYSIS COMPLETE!
==================================================
{
  "video_duration": "00:55",
  "scene_changes": 2,
  "subjects_detected": [
    "An adult cheetah, resting in a semi-upright position. Its coat is covered in distinctive black spots on a tan background.",
    "A young cheetah cub, significantly smaller than the adult, following behind.",
    "An adult cheetah, distinguished by its slender build, spotted coat, and a tear-like streak running down its face.",
    "A cheetah cub, positioned in the grass to the right of the adult. It is also spotted, but with fluffier fur and a less distinct pattern. It appears to be sniffing or exp
    "An adult cheetah, partially visible on the left of the frame, showing the lower part of the body and legs.",
    "A cheetah cub is walking towards the left of the frame. It is light brown with faint spots, indicating its young age.",
    "Adult Cheetah: Standing tall, with a distinctive spotted coat, and a long, slightly curved tail with black bands near the end. It looks directly at the camera, alert and
    "sheep",
    "An adult cheetah, sitting and partially facing the camera. It has characteristic black spots on its golden fur and a slightly open mouth.",
    "A cheetah: the back legs of an adult cheetah standing behind the cub, also covered in distinctive spots.",
    "A cheetah cub, small and covered in spots, lying in the grass.",
    "giraffe",
    "Baboon (smaller, brown furred animal, appears to be juvenile or smaller species)",
    "A young cheetah sitting on or beside a freshly killed prey (possibly a small mammal) located on the right side of the frame. The cheetah cub appears smaller and has simi
    "A full-grown cheetah standing on the left side of the frame, with a slender build and distinctive black spots on its tan fur. It is facing towards the right side of the
    "A cheetah cub, possessing a similar spotted coat to its parent, indicative of its young age.",
    "A rabbit, attempting to flee from the lion cub.",
    "A dead gazelle or similar prey animal: lying on the ground, being manipulated by the cub.",
    "A cheetah, standing, observing the interaction between the other subjects.",
    "A cheetah cub, lying prone in the grass, also covered in the characteristic spotted coat.",
    "A young cheetah cub with distinctive spotted fur is standing over its prey."
```

Intelligent Video Conversation Agent

This cell creates an AI assistant specifically trained to discuss the analyzed video.

Capabilities:

- **Content-Aware**: Adapts its expertise to the type of video (wildlife, sports, film,city data, etc.)
- **Context-Aware**: Maintains conversation history for natural follow-up questions
- **Detail-Oriented**: Can discuss specific moments, subjects, or elements
- **Educational**: Provides insightful analysis based on video content

The agent is given comprehensive access to:

- The frame-by-frame analysis data
- Object detection results
- Scene changes and timelines
- All detected subjects, activities, and contexts

This enables natural language exploration of the video content.

[11]
```
class VideoIntelligenceBot:
    def __init__(self, gemini_api_key, analysis_data, analyzer_instance):
        self.model = genai.GenerativeModel("gemini-2.0-flash-exp")
        self.analysis_data = analysis_data
        self.analyzer = analyzer_instance
        self.conversation_history = []

    def get_context(self):
        """Prepare detailed context for conversation"""
```

```
            Prepare detailed context for conversation
        context = f"""
VIDEO ANALYSIS SUMMARY:
Video Duration: {self.analysis_data['video_duration']}
Scene Changes: {self.analysis_data['scene_changes']}
Detected Genres: {', '.join(self.analysis_data.get('detected_genres', ['Unknown']))}

SUBJECTS DETECTED: {', '.join(self.analysis_data['subjects_detected'])}
ACTIVITIES OBSERVED: {', '.join(self.analysis_data['activities_observed'])}
INTERACTIONS: {', '.join(self.analysis_data['interactions'])}
SETTINGS: {', '.join(self.analysis_data['settings'])}
EMOTIONAL TONES: {', '.join(self.analysis_data['emotional_tones'])}
NOTABLE ELEMENTS: {', '.join(self.analysis_data['notable_elements'])}
TECHNICAL ELEMENTS: {', '.join(self.analysis_data['technical_elements'])}

DETAILED TIMELINE:
{chr(10).join(self.analysis_data['timeline'])}

FRAME-BY-FRAME ANALYSIS:
"""

        for i, analysis in enumerate(self.analyzer.frame_analyses):
            context += f"\n--- Analysis {i+1} at {self.analyzer.format_time(analysis['timestamp'])} ---\n"
            gemini_data = analysis['gemini_analysis']
            context += f"Subjects: {', '.join(gemini_data.get('subjects', []))}\n"
            context += f"Activities: {', '.join(gemini_data.get('activities', []))}\n"
            context += f"Setting: {gemini_data.get('setting', 'Unknown')}\n"
            context += f"Emotional Tone: {gemini_data.get('emotional_tone', 'Unknown')}\n"
            context += f"Summary: {gemini_data.get('summary', 'No summary')}\n"

            yolo_objects = [obj['object'] for obj in analysis['yolo_objects']]
            context += f"YOLO Objects: {', '.join(set(yolo_objects))}\n"

        return context

    def ask(self, question):
        """Ask anything about the analyzed video"""
        context = self.get_context()

        prompt = f"""
You are an intelligent video analysis assistant with expertise in all types of video content. You have performed a detailed analysis of a video and have comprehensive informat

ANALYSIS CONTEXT:
{context}

CONVERSATION HISTORY:
{self.format_history()}

USER QUESTION: {question}

Provide a detailed, insightful answer based on the video analysis. Adapt your expertise to the content type.
If the video appears to be:
- Nature/wildlife: Focus on animal behavior, biology, ecosystem dynamics
- Sports: Focus on techniques, performance analysis, strategy
- Film/Entertainment: Focus on narrative, cinematography, character dynamics
- Educational: Focus on key concepts, learning elements
- News/Documentary: Focus on events, subjects, context

If the question cannot be answered from the analysis data, say so clearly.
"""

        try:
            response = self.model.generate_content(prompt)
            answer = response.text

            # Store conversation
            self.conversation_history.append({"question": question, "answer": answer})
            return answer

        except Exception as e:
            return f"Encountered an error: {e}"

    def format_history(self):
        """Format recent conversation history"""
        if not self.conversation_history:
            return "No previous conversation."

        history = ""
        for exchange in self.conversation_history[-3:]:
            history += f"Q: {exchange['question']}\nA: {exchange['answer']}\n\n"
        return history

# Initialize the video intelligence chatbot
video_chat = VideoIntelligenceBot(GEMINI_API_KEY, analysis_summary, analyzer)
print("Video Intelligence Chatbot ready!")
```

```
Video Intelligence Chatbot ready!
```

Interactive Video Exploration

Now you can have a conversation about the video! Ask questions like:

- **Content Questions**:
  - "What are the main subjects in this video?"
  - "What activities are taking place?"
  - "How would you describe the setting?"
- **Technical Questions**:

- "What cinematography techniques are used?"
- "How many scene changes were detected?"
- "What's the lighting like throughout the video?"

- **Analytical Questions**:
  - "What is the emotional tone of this video?"
  - "How do the subjects interact with each other?"
  - "What's the narrative structure of this content?"
- **Timeline Questions**:
  - "What happens at 00:35?"
  - "Describe the sequence of events in the first minute"
  - "When does the first scene change occur?"

Type your questions below or 'exit' to quit the chat interface.

[18]
```
print("INTELLIGENT VIDEO-CHAT MODE")
print("Hi I am Tanmay captain of this team, Ask me about the analyzed video!")
print("\nType your questions below:")

def chat_interface():
    while True:
        print("="*150)
        question = input("\n You: ")
        if question.lower() in ['exit', 'quit', 'stop']:
            print(" Thanks for exploring the video analysis with me! Goodbye!")
            break

        if question.strip():
            answer = video_chat.ask(question)
            print(f"\n Video Expert: {answer}")
        else:
            print("Please ask a question about the video!")

chat_interface()
```

```
INTELLIGENT VIDEO-CHAT MODE
Hi I am Tanmay captain of this team, Ask me about the analyzed video!

Type your questions below:
==========================================================================================================================================================

 You: what is the weather in the jungle?

 Video Expert: The video analysis does not provide sufficient information to determine the weather in a jungle. The video shows a savanna or grassland environment, not a jung

==========================================================================================================================================================

 You: Describe the main characters and their relationships? keep it as a shot and straight response

 Video Expert: Adult cheetah (provider, protector, teacher) and cub (learner, dependent). Strong mother-cub bond. Predator-prey dynamics with gazelles, jackals etc. Unusual,

==========================================================================================================================================================

 You: what is the cub learning ?

 Video Expert: Based on the video analysis, the cheetah cub is learning several key survival skills:

*   **Hunting and Predation:** The cub is observed interacting with and feeding on prey (gazelle, jackal), suggesting it's learning how to manipulate, consume, and potentiall

*   **Scavenging:** At [00:40] the cheetah is carrying an unidentified object - possibly something scavenged. This may indicate the cub learns the benefits of scavenging and

*   **Predator-Prey Dynamics:** The cub's proximity to the adult during hunting/feeding highlights the predator-prey relationship and its role in the ecosystem.

*   **Environmental Awareness:** The cub's exploration of the grassland environment ([00:50]) suggests it is learning to identify scents, navigate its habitat, and become fam

*   **Survival Skills and Alertness:** The cub learns how to stay alert, and how to survey the environment.

*   **Social Behavior:** By following the adult cheetah, the cub is learning social behaviors, communication, and the importance of familial bonds. The adult's protective beh

*   **Guarding prey** The cub is also learning how to guard its kill and ward off other possible threats or scavengers.

The adult cheetah is actively involved in the cub's learning process, whether through direct instruction, demonstration, or simply providing a safe environment for exploratio

==========================================================================================================================================================

 You: describe the  cinematic techniques are used in this video

 Video Expert: Based on the video analysis, here are the cinematic techniques observed in this wildlife documentary footage:

*   **Natural Lighting:** The video relies heavily on natural lighting, creating a realistic and immersive viewing experience. The use of sunlight casts shadows and highlight

*   **Medium Shots:** Medium shots are frequently employed to capture both the cheetahs and their immediate surroundings, providing context for their behavior and interaction

*   **Shallow Depth of Field:** A shallow depth of field is used to keep the cheetahs in sharp focus while softening the background elements. This technique helps to draw the

*   **Telephoto Lens:** The use of a telephoto lens is evident in some shots, compressing the space and bringing the subjects closer to the viewer. This technique allows for

*   **Composition:** The composition of the shots often focuses on the relationship between the adult cheetah and its cub, highlighting their familial bond and the transfer o

*   **Framing:** The framing of the shots often centers on the animals and their interactions, such as the cub manipulating prey or the adult standing guard. The use of natur

*   **Static Shots:** The cinematography involves relatively static shots, allowing viewers to observe the cheetahs' behavior without excessive camera movement. This approach
```

Detailed Analysis Results & Insights

This final section provides:

1. **Comprehensive Results Display**
    - Complete content analysis organized by category
    - Chronological timeline of the entire video
    - Technical metrics about the analysis process
2. **Automated Insights**
    - The system detects the content type and generates relevant questions
    - Questions are automatically adapted to the video genre
    - Provides quick summary insights without manual prompting
3. **Results Export**
    - All analysis data will be saved to a JSON file for further processing
    - Can be used for creating custom visualizations or reports
    - Enables integration with other systems or applications

pipeline finished here!

```
[14] def display_detailed_results():
         print("DETAILED ANALYSIS RESULTS")
         print("="*50)

         print(f"\n VIDEO INFORMATION:")
         print(f"Duration: {analysis_summary['video_duration']}")
         print(f"Analysis points: {analysis_summary['analysis_points']}")
         print(f"Scene changes: {analysis_summary['scene_changes']}")
         print(f"Detected genres: {', '.join(analysis_summary.get('detected_genres', ['Unknown']))}")

         print(f"\n CONTENT FINDINGS:")
         print(f"Subjects detected: {', '.join(analysis_summary['subjects_detected'])}")
         print(f"Activities observed: {', '.join(analysis_summary['activities_observed'])}")

         print(f"\n CONTEXT & DETAIL:")
         print(f"Interactions: {', '.join(analysis_summary['interactions'])}")
         print(f"Notable elements: {', '.join(analysis_summary['notable_elements'])}")
         print(f"Technical elements: {', '.join(analysis_summary['technical_elements'])}")

         print(f"\n SETTING & MOOD:")
         print(f"Settings: {', '.join(analysis_summary['settings'])}")
         print(f"Emotional tones: {', '.join(analysis_summary['emotional_tones'])}")

         print(f"\n TIMELINE:")
         for event in analysis_summary['timeline']:
             print(f"  {event}")

         # Export to JSON file
         with open('video_analysis_results.json', 'w') as f:
             json.dump({
                 'video_name': video_path,
                 'analysis_summary': analysis_summary,
                 'detailed_frames': analyzer.frame_analyses
             }, f, indent=2)

         print(f"\n Results exported to 'video_analysis_results.json'")

     # Display the results
     display_detailed_results()
```

```
DETAILED ANALYSIS RESULTS
==================================================

 VIDEO INFORMATION:
Duration: 00:55
Analysis points: 12
Scene changes: 2
Detected genres: wildlife documentary, Nature Documentary/Wildlife, Nature Documentary, Wildlife/Nature Documentary, Wildlife documentary, Wildlife Documentary

 CONTENT FINDINGS:
Subjects detected: An adult cheetah, resting in a semi-upright position. Its coat is covered in distinctive black spots on a tan background., A young cheetah cub, significantly
Activities observed: The adult cheetah seems to be on alert, possibly surveying the surroundings., The adult cheetah is standing nearby, observing the cub., The cheetah cub is

 CONTEXT & DETAIL:
Interactions: There is a clear predator-prey relationship between the cheetah and the unseen prey., Predator-prey relationship: The lion cub is hunting the rabbit, while the ch
Notable elements: The intergenerational dynamic of a cheetah and its cub., The distinctive coat patterns of the cheetahs provide excellent camouflage in their environment., The
Technical elements: Natural lighting creates a sunny and bright atmosphere. The focus is sharp on the cub, with a shallower depth of field around the adult., Shallow depth of f

 SETTING & MOOD:
Settings: The scene is set in a grassy environment, likely a savanna or grassland habitat. There is tall grass and some bare earth/dirt visible in the background. The time of d
Emotional tones: The mood is tense due to the unusual encounter. There's a sense of curiosity and potential conflict in the scene. The emotional content is driven by the uncert

 TIMELINE:
  00:00: An adult cheetah carries prey in its mouth as a cheetah cub follows closely behind through the tall grasses of the African savanna.
  00:05: An adult cheetah and its cub are captured in a medium shot as they traverse a golden grassland, likely as part of a documentary showcasing their natural habitat and be
  00:10: In a savanna setting, a cheetah is eating its prey while a cub approaches, suggesting a family dynamic and the raw reality of survival in the wild.
  00:15: A cheetah cub manipulates a dead gazelle carcass on an African savanna, with its parent nearby, likely demonstrating or learning hunting skills.
  00:20: A frame shows an adult cheetah standing guard over a young cheetah feeding on its kill in a grassy savanna setting, reflecting a moment of survival and maternal care i
  00:25: A cheetah observes as a young lion cub chases a rabbit across a grassy field, potentially a training scenario or wildlife demonstration.
  00:30: A young cheetah cub stands over a dead jackal in an open grassland setting, depicting a key moment in a potential wildlife documentary showcasing predator-prey dynamic
  00:35: In a savanna environment, an adult cheetah and its cub are feeding on their prey, depicting the natural behavior and dynamics of a wild predator-prey relationship.
  00:40: A cheetah carries an object while a baboon walks alongside in a dry savanna, creating a tense and curious scene of potential conflict or unique interaction.
  00:45: An adult cheetah rests watchfully beside its cub in a grassy savanna, depicting a peaceful and tender moment in the wild.
  00:50: An adult cheetah rests while its cub explores the surrounding grass, showcasing a peaceful moment of wildlife behavior in a natural habitat.
  00:55: A cheetah cub rests in the grass next to its mother, showcasing a moment of peace and maternal care in a savanna environment.

 Results exported to 'video_analysis_results.json'
```

Project Summary and Applications

Key Technical Achievements

- **Multi-modal analysis** combining computer vision and large language models
- **Genre-adaptive questioning** that tailors insights to content type
- **Human-like understanding** of complex visual scenes and narratives
- **Interactive exploration** through natural conversation

Potential Applications

- **Media Analysis**: Automated content tagging, classification, and summarization
- **Education**: Enriched video learning experiences with interactive Q&A
- **Accessibility**: Detailed descriptions of video content for visually impaired users
- **Research**: Analyzing behavior, interactions, and patterns in video data
- **Creative Industries**: Insights into cinematography, narrative structure, and visual techniques

Future Enhancements

- Audio analysis integration
- Emotion recognition from facial expressions
- Action recognition for complex behaviors
- Character/subject tracking across scenes
- Custom domain-specific analysis models
- Multi model response, adding relevent frames to the response to support the answer

Thank you for exploring our System!