Sistema de Detección de Conductas Sospechosas en Exámenes en Línea

Desarrollado por: Fernando Imre Capobianco Török

Cómo desarrollar un sistema automatizado que detecte posibles intentos de copia en exámenes virtuales mediante el análisis de video y audio?

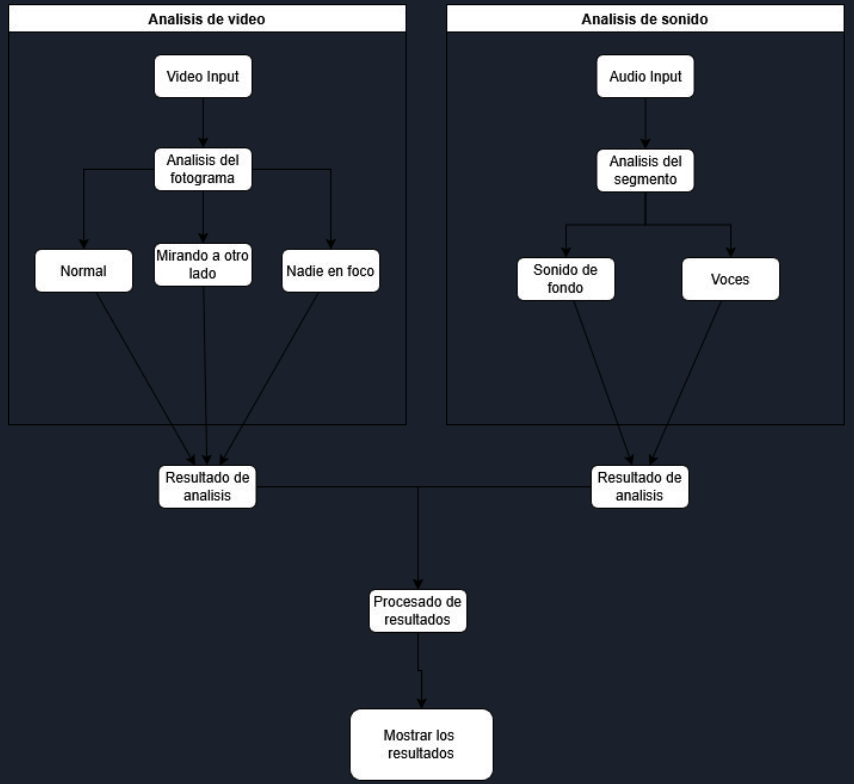
Desarrollar una herramienta basada en IA para identificar conductas sospechosas en exámenes en línea

- Entrenar modelos de visión por computadora para clasificar acciones en video
- Entrenar modelos de procesamiento de audio para detectar voz humana
- Implementar interfaz gráfica para visualizar alertas

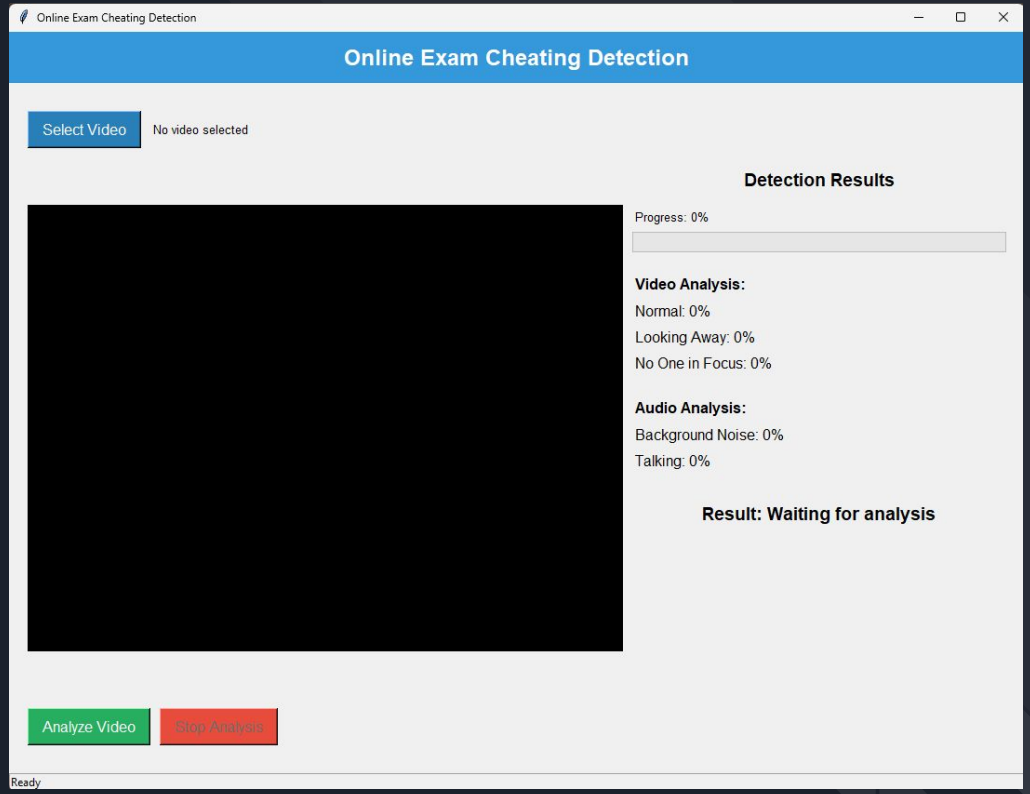
Tecnologia Implementadas

- TensorFlow Lite: Ejecución de modelos de aprendizaje automático
- OpenCV y Pillow: Captura y procesamiento de video
- MoviePy: Extracción de audio
- Tkinter: Interfaz gráfica de usuario
- Teachable Machine: Entrenamiento de modelos

Arquitectura



Funcionamiento de la interfaz



Algoritmos más importantes

Analisis de video

- Conversión de color: Transforma la imagen de BGR a RGB para compatibilidad con el modelo.
- Preprocesamiento: Redimensiona y normaliza la imagen para el formato que requiere el modelo.
- Ejecución de inferencia: Carga el fotograma en el intérprete TensorFlow Lite y ejecuta el modelo.
- Interpretación de resultados: Obtiene la clase predicha (comportamiento normal, mirada desviada o ausencia) y su nivel de confianza.

```python
def predict_video_frame(self, frame):
    # Convert BGR to RGB (if needed)
    if frame.shape[2] == 3:
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Preprocess the image
    input_data = self.preprocess_video_frame(frame)

    # Set input tensor and run inference
    self.video_interpreter.set_tensor(self.video_input_details[0]['index'], input_data)
    self.video_interpreter.invoke()

    # Get output tensor and predicted "class"
    output_data = self.video_interpreter.get_tensor(self.video_output_details[0]['index'])
    predicted_class = np.argmax(output_data)
    confidence = output_data[0][predicted_class]

    return predicted_class, confidence// put your code here
```

Analisis de audio

- Segmentación: Divide el audio en fragmentos pequeños para análisis.
- Normalización: Convierte los valores de audio a un rango apropiado para el modelo.
- Inferencia: Ejecuta cada segmento a través del modelo de clasificación de audio.
- Conteo de clases: Registra cuántos segmentos contienen voz humana vs. ruido de fondo.

```python
def analyze_audio(self, audio_path):
    # Load and process audio file
    # For each segment:
    for i in range(min(100, num_segments)):
        # Extract segment and preprocess
        segment = audio_data[start:end]
        segment = segment.astype(np.float32) / (2**(8 * sample_width - 1))

        # Reshape and run inference
        input_data = np.reshape(segment, input_shape)
        self.audio_interpreter.set_tensor(self.audio_input_details[0]['index'], input_data)
        self.audio_interpreter.invoke()

        # Get predictions
        output_data = self.audio_interpreter.get_tensor(self.audio_output_details[0]['index'])
        predicted_class = np.argmax(output_data)
        class_counts[predicted_class] += 1
```

Lógica de Detección

- Cálculo de porcentajes: Determina qué proporción del tiempo el estudiante mostró cada comportamiento.
- Aplicación de umbrales: Considera sospechoso si el comportamiento normal es <70% del tiempo.
- Detección combinada: Evalúa señales de video y audio en conjunto para mayor precisión.
- Clasificación final: Categoriza el examen como normal o sospechoso, especificando el tipo de alerta.

```python
def update_final_results(self, video_class_counts, audio_class_counts, total_frames):
    # Calculate percentages for video and audio
    normal_pct = (video_class_counts[0] / total_frames) * 100
    looking_away_pct = (video_class_counts[1] / total_frames) * 100
    no_one_pct = (video_class_counts[2] / total_frames) * 100

    total_audio_segments = sum(audio_class_counts.values())
    talking_pct = (audio_class_counts[1] / total_audio_segments) * 100

    # Determine if cheating was detected
    is_video_cheating = normal_pct < 70
    is_audio_cheating = talking_pct > 35

    if is_video_cheating and is_audio_cheating:
        result = "SUSPICIOUS - Cheating detected (Video & Audio)"
    elif is_video_cheating:
        result = "SUSPICIOUS - Cheating detected (Video)"
    elif is_audio_cheating:
        result = "SUSPICIOUS - Cheating detected (Audio)"
    else:
        result = "NORMAL - No cheating detected"
```

Demostración de la Interfaz

¡Gracias!