🎓 Tech Challenge - Pós-Tech - IA For Devs - FIAP

📹 Fase 4 - Análise de vídeo com IA

👥 1. Alunos:

- André Mattos - RM358905
- Aurelio Thomasi Jr - RM358104
- Leonardo Ramires - RM358190
- Lucas Arruda - RM358628
- Pedro Marins - RM356883

📋 2. Evidências do projeto

- Link para o repositório:Repositorio Git
- Link para o vídeo de apresentação: [Video Apresentação]
- Vosk Model: Vosk Model

📚 3. Bibliotecas utilizadas

- Principais bibliotecas:
  - **OpenCV (cv2)**: Biblioteca utilizada para processamento de vídeo, detecção de rostos e manipulação de imagens.
  - **DeepFace**: Biblioteca utilizada para análise de emoções faciais (feliz, triste, etc).
  - **MediaPipe**: Biblioteca utilizada para detecção de movimentos(pose corporal, movimentos das mãos, etc).
  - **YOLO**: Biblioteca alternativa que foi utilizada para detectar faces e classificar emoções.
- Bibliotecas de suporte:
  - **Dlib**: Biblioteca base para o face_recognition, utilizada para detecção e codificação de rostos.

- **Tensorflow**: Dependência do DeepFace para análise de emoções.
- **Vosk**: Modelo utilizada para transcrição de áudio do vídeo para texto.
- **Pandas**: Biblioteca utilizada para geração de relatórios e análise dos dados coletados.
- **NumPy**: Biblioteca utilizada para operações matemáticas e manipulação de arrays.
- **tqdm**: Biblioteca utilizada para exibir barras de progresso durante o processamento do vídeo que está sendo analisado.

💻 4. Instalar Dlib e Tensorflow (Windows)

Durante o desenvolvimento do projeto, foi necessário instalar o Dlib e o Tensorflow para a utilização de CUDA, para processar os vídeos com GPU e consequentemente melhorar o desempenho do processamento. No final desta documentação, será apresentado o passo a passo para instalar o Dlib e o Tensorflow para o ambiente Windows (ambiente de desenvolvimento utilizado).

CUDA: É uma biblioteca de software utilizada em hardware de computação gráfica da empresa NVIDIA, que permite a utilização de GPUs para acelerar o processamento de cálculos matemáticos (Por exemplo, matrizes, cálculos de IA, etc).

📝 6. Descrição

Este Tech Challenge tem como objetivo de criar uma aplicação que utilize análise de vídeo com IA, para detectar os seguintes eventos: - Reconhecimento facial: Identificar e marcar pessoas no vídeo. - Análise de expressões emocionais: Identificar e analise expressões dos rostos identificados. - Detecção de atividades: Detectar e categorizar atividades sendo realizadas no vídeo. - Geração de resumo: Um resumo automático das principais atividades e emoções detectadas no vídeo.

Após as detecções, os sistema irá gerar automaticamente um relatório com as principais atividades e emoções detectadas no vídeo. O relatório deve incluir: - Total de frames analisados - Número de anomalias detectadas

👤 7. Detecção de rostos - reconhecimento facial

Esta parte do projeto foi desenvolvida utilizando a bibliotca **OpenCV** para realizar o processamento. Os seguintes passos são realizados neste processo:

1. Carregar o modelo de detecção de rostos.

2. Carregar o vídeo.
3. Processar o vídeo frame a frame.
4. Detectar rostos no frame.
5. Desenhar retângulos ao redor dos rostos detectados.
6. Identificar nome de pessoas de acordo com imagens de referência que estão salvas no diretório `images`.
7. Salvar o frame com os retângulos desenhados.
8. Atualizar o relatório com os dados coletados.

7.1 Imagens de referência

As imagens de referência foram salvas no diretório `images` e foram utilizadas para identificar as pessoas no vídeo. As seguintes imagens foram utilizadas:

- **Ann** (01Ann_A01.png)
- **Brunna** (01Brunna_B01.png)
- **Charles** (01Charles_C01.png)
- **Danielle** (01Danielle_D01.png)
- **Ed** (02Ed_E01.png)
- **Faith** (04Faith_F01.png)
- **Garth** (05Garth_G01.png)
- **Harry** (06Harry_H01.png)
- **Ivy** (07Ivy_I01.png)
- **John** (08John_J01.png)
- **Kay** (12Kay_K01.png)
- **Lana** (13Lana_L01.png)
- **Mark** (14Mark_M01.png)
- **Noah** (15Noah_N01.png)
- **Oswald** (17Oswald_O01.png)
- **Paula** (17Paula_P01.png)
- **Rita** (17Rita_R01.png)
- **Saul** (17Saul_S01.png)
- **Thor** (18Thor_T01.png)

7.2 Processamento do vídeo e parâmetros

A função `face_detection_and_recognition` é responsável por realizar o reconhecimento facial no vídeo. O processo consiste em:

1. **Carregamento de Dados**:
    - Carrega as imagens de referência do diretório especificado.
    - Inicializa o vídeo de entrada e cria o arquivo de saída.
    - Prepara as estruturas de dados para armazenar os resultados.
2. **Processamento Frame a Frame**:
    - Para cada frame do vídeo:
        - Converte o frame para RGB (necessário para o `face_recognition`).
        - Detecta rostos e gera codificações faciais.
        - Compara com as imagens de referência.
        - Desenha retângulos e nomes ao redor dos rostos identificados.
        - Salva os resultados para análise posterior.
3. **Geração de Relatório**:
    - Cria um arquivo CSV com os resultados.
    - Gera um resumo da análise com estatísticas.
4. **Anomalias**:
    - Se o nome da pessoa não for encontrado nas imagens de referência, o sistema irá marcar a pessoa como “Anônimo”.

7.2.1 Parâmetros da Função

Parâmetro	Tipo	Descrição
`images_path`	str	Caminho para o diretório contendo as imagens de referência
`video_in_path`	str	Caminho do arquivo de vídeo de entrada
`video_out_path`	str	Caminho onde será salvo o vídeo processado

7.2.2 Parâmetros Internos

Parâmetro	Valor Padrão	Descrição
`number_of_times_to_upsample`	5	Número de vezes que a imagem é redimensionada para detectar rostos menores
`model`	“cnn”	Modelo de detecção facial (“cnn” para GPU, “hog” para CPU)
`num_jitters`	40	Número de amostragens para gerar a codificação facial
`encoding_model`	“large”	Modelo de codificação facial (“large” para 128 pontos, “small” para 5 pontos)

7.2.3 Saídas

1. **Vídeo Processado**:
    - Arquivo MP4 com os rostos identificados.
    - Retângulos coloridos ao redor dos rostos.
    - Nomes das pessoas identificadas.
2. **Arquivo CSV**:
    - Frame ID.
    - Nome da pessoa identificada.
    - Timestamp do frame.
3. **Resumo da Análise**:
    - Total de frames processados.
    - Estatísticas de detecção por pessoa.

- Tempo total de processamento.

😊 8. Detecção de emoções - expressões faciais

Esta parte do projeto foi desenvolvida utilizando a biblioteca **DeepFace** para realizar a análise de emoções. O processo consiste em:

1. **Carregamento de Dados**:
   - Inicializa o vídeo de entrada.
   - Prepara o arquivo de saída para o vídeo processado.
   - Configura as estruturas de dados para armazenar os resultados.
2. **Processamento Frame a Frame**:
   - Para cada frame do vídeo:
     - Detecta rostos usando OpenCV (DNN ou Haar Cascade).
     - Analisa as emoções de cada rosto detectado usando DeepFace e escreve o nome da emoção que foi detectada.
     - Desenha retângulos ao redor dos rostos identificados.
     - Salva os resultados para análise posterior.
3. **Geração de Relatório**:
   - Cria um arquivo CSV com os resultados
   - Gera um resumo da análise com estatísticas de emoções

8.1 Parâmetros da Função

Parâmetro	Tipo	Descrição
`video_in_path`	str	Caminho do arquivo de vídeo de entrada
`video_out_path`	str	Caminho onde será salvo o vídeo processado

8.2 Parâmetros Internos

Parâmetro	Valor Padrão	Descrição
`actions`	[‘emotion’]	Lista de ações a serem analisadas pelo DeepFace
`detector_backend`	‘centerface’	Backend de detecção facial (‘opencv’, ‘mtcnn’, ‘skip’, ‘mediapipe’ ou ‘centerface’)
`enforce_detection`	False	Se deve forçar a detecção mesmo com baixa confiança
`anti_spoofing`	False	Se deve verificar se o rosto é real ou uma foto

8.3 Emoções Detectadas

O sistema é capaz de detectar as seguintes emoções: - **Feliz** (happy) - **Triste** (sad) - **Neutro** (neutral) - **Surpreso** (surprise) - **Com medo** (fear) - **Irritado** (angry) - **Desconhecido** (unknown) - quando não é possível detectar a emoção

8.4 Saídas

1. **Vídeo Processado**:
    - Arquivo MP4 com os rostos e emoções identificados.
    - Retângulos coloridos ao redor dos rostos.
    - Emoção detectada para cada rosto.
2. **Arquivo CSV**:
    - Frame ID.
    - Emoções detectadas (até 4 emoções por frame).
    - Timestamp do frame.
3. **Resumo da Análise**:
    - Total de frames processados.

- Estatísticas de cada emoção detectada.
- Tempo total de processamento.

🎙️ 9. Transcrição do vídeo

Esta parte do projeto foi desenvolvida utilizando as bibliotecas **MoviePy** e **Vosk** para realizar a transcrição do áudio do vídeo. Isso não é um requisito p ara o projeto, mas foi uma opção considerada para o desenvolvimento porque é uma análise útil e faz parte da fase atual da Pos-Tech. O processo consiste em:

1. **Extração do Áudio**:
   - Carrega o arquivo de vídeo usando MoviePy.
   - Extrai a faixa de áudio do vídeo.
   - Salva o áudio em formato WAV com qualidade CD (44.1kHz, 16-bit, mono).
2. **Processamento do Áudio**:
   - Carrega o arquivo de áudio extraído.
   - Converte o áudio para o formato adequado para reconhecimento de fala.
   - Utiliza o modelo Vosk para transcrição offline.
3. **Geração da Transcrição**:
   - Realiza o reconhecimento de fala.
   - Salva o texto transcrito em um arquivo.
   - Exibe o progresso durante o salvamento.

9.1 Parâmetros da Função

Parâmetro	Tipo	Descrição
`video_in_path`	str	Caminho do arquivo de vídeo de entrada
`audio_out_path`	str	Caminho onde será salvo o arquivo de áudio extraído
`text_out_path`	str	Caminho onde será salva a transcrição em texto

9.2 Parâmetros Internos

Parâmetro	Valor Padrão	Descrição
`fps`	44100	Taxa de amostragem do áudio (Hz)
`nbytes`	2	Profundidade de bits (2 = 16-bit)
`codec`	‘pcm_s16le’	Codec de áudio (WAV 16-bit)
`ffmpeg_params`	[“-ac”, “1”]	Parâmetros para forçar saída mono

9.3 Requisitos

1. **Modelo Vosk**:
    - É necessário baixar e instalar o modelo de linguagem Vosk.
2. **Dependências**:
    - **MoviePy**: Biblioteca utilizada para manipulação de vídeo e áudio.
    - **SpeechRecognition**: Biblioteca utilizada para interface com o Vosk.
    - **FFmpeg**: Biblioteca utilizada para processamento de áudio.

9.4 Saídas

1. **Arquivo de Áudio**:
    - Formato WAV
    - Qualidade CD (44.1kHz)
    - Áudio mono
    - 16-bit de profundidade
2. **Arquivo de Transcrição**:
    - Formato texto (.txt)
    - Codificação UTF-8
    - Texto transcrito do áudio
    - Salvo em chunks para melhor performance

3. **Feedback**:
    - Barra de progresso durante o processamento
    - Mensagens de status no console
    - Transcrição exibida no terminal

📊 10. Modelo YOLO

Esta seção tem como objetivo apresentar uma solução alternativa para a detecção de faces e classificação de emoções, utilizando o modelo YOLO.

O modelo YOLO (You Only Look Once) é um modelo que é capaz de detectar objetos em tempo real. Ele é capaz de detectar faces e classificar emoções com uma precisão muito alta. Este modelo foi utilizado para comparar com a solução atual utilizada no projeto, com intuito de validar soluções alternativas que podem ser mais eficientes para o problema proposto.

👤 10.1 Detecção de faces

A detecção de faces foi realizada utilizando o modelo YOLO11, que traz melhorias de desempenho, precisão, flexibilidade e eficiência em Visão Computacional. O processo de detecção de faces foi realizado no arquivo `recognize_expression_yolo.py`.

🔄 **Passos da Implementação:**

1. **Carregamento e Configuração do Modelo**:
    - Utiliza o modelo YOLOv11 especificamente treinado para detecção de faces (`yolov11l-face.pt`)
    - O modelo é carregado do diretório `./doc/model`
2. **Processo de Detecção Facial** (função `detectar_pessoas`):
    - Recebe um frame e o processa usando o modelo YOLO
    - Para cada face detectada:
        - Extrai as coordenadas da face (x1, y1, x2, y2)
        - Desenha retângulos ao redor dos rostos detectados
3. **Análise de Emoções** (função `analisar_emocao`):
    - Utiliza DeepFace em conjunto com YOLO

- Redimensiona as faces detectadas para `224x224 pixels`
- Analisa emoções usando `MTCNN` como backend de detecção (está hardcoded no código)
- Retorna a emoção dominante para cada face

4. **Processamento de Vídeo** (função `processar_video`):
   - Processa o vídeo frame a frame com o seguinte fluxo:
     - Lê as propriedades do vídeo (largura, altura, fps, total de frames)
     - Cria um escritor de vídeo de saída com codec `MP4`
     - Processa a cada `5 frames` para detecção facial e análise de emoções
     - Utiliza processamento paralelo para análise de emoções
     - Desenha retângulos ao redor das faces detectadas e adiciona texto com a emoção detectada nos frames
     - Salva os frames processados no vídeo de saída
5. **Processamento de Resultados**:
   - Salva resultados em arquivo CSV com colunas:
     - frame_id
     - emotions_1 até emotions_4 (até 4 emoções por frame)
   - Gera análise resumida em `summary_analysis.txt`
   - **Restaura áudio do vídeo original para o vídeo de saída**

📊 **Resultados Esperados:**

1. **Saída Visual**:
   - Um arquivo de vídeo processado (`tc4_video_fe_yolo.mp4`) contendo:
     - Retângulos ao redor das faces detectadas
     - Texto com a emoção detectada para cada face
     - Áudio original preservado do vídeo de entrada
2. **Saída de Dados**:
   - Arquivo CSV (`tc4_video_fe_yolo.mp4.csv`) contendo:
     - Resultados de detecção de emoções frame a frame

        - Até 4 emoções por frame
        - Timestamps para cada detecção

3. **Resumo da Análise**:
    - Arquivo de resumo (`summary_analysis.txt`) com:
        - Total de aparições de cada emoção
        - Análise de segmentos de emoção (mínimo 5 frames)
        - Limiar de pausa de 10 frames entre emoções

⚡ **Características Principais:**

1. **Otimizações de Desempenho**:
    - Processa a cada `5 frames` para reduzir carga computacional
    - Utiliza processamento paralelo para análise de emoções
    - Reutiliza resultados de detecção facial entre frames
2. **Detecção de Emoções**:
    - Combina a eficiente detecção facial do YOLO com a análise de emoções do DeepFace
    - Pode detectar múltiplas emoções por frame
    - Lida de maneira inteligente com casos onde a detecção de emoção falha

Esta implementação fornece uma solução robusta para detecção facial e de emoções, aproveitando as capacidades eficientes de detecção facial do YOLO enquanto utiliza o DeepFace para análise detalhada de emoções. O sistema foi projetado para ser tanto eficiente (através de pulo de frames e processamento paralelo) quanto abrangente em sua análise de expressões faciais em conteúdo de vídeo.

⚖️ 10.2 Comparação entre YOLO e a solução atual

Abaixo está uma tabela comparativa entre a solução atual e a solução utilizando o modelo YOLO.

Característica	Solução Atual	Solução YOLO
Detecção de Faces	Usa OpenCV com DNN	Usa YOLOv11 especializado
Precisão de Detecção	Média	Alta (modelo especializado em faces)
Velocidade de Processamento	Processa todos os frames	Processa a cada 5 frames (mais eficiente)
Detecção de Emoções	DeepFace direto	YOLO + DeepFace em paralelo
Quantidade de Emoções Detectadas	75 detecções totais	154 detecções totais
Processamento Paralelo	Não implementado	Implementado para análise de emoções
Uso de GPU	Limitado	Otimizado para GPU
Consumo de Memória	Alto	Moderado (devido ao processamento em lotes)
Flexibilidade	Modelo fixo	Diferentes tamanhos de modelo disponíveis (n, s, m, l, x)

Observações: - A solução YOLO detectou mais de **2x** emoções em comparação com a solução atual - O processamento paralelo na solução YOLO permite análise mais rápida - A solução YOLO é mais eficiente em termos de recursos computacionais - A precisão da detecção facial é superior na solução YOLO devido ao modelo especializado - O YOLO consegue detectar mais faces, em condições de baixa luminosidade e também quando o rosto está parcialmente bloqueado ou com rosto inclinado.