UNIVERSIDADE DO MINDELO

DEPARTAMENTO DE ENGENHARIA E RECURSOS DO MAR

CURSO DE LICENCIATURA EM

ENGENHARIA INFORMÁTICA E SISTEMAS COMPUTACIONAIS

ATIVIDADE PRÁTICA

ANO LETIVO 2023/2024 – 4ºANO

TEMA: Sistema de Classificação de Alunos em Tempo Real com Transfer Learning.

Discente: Anifa Pinheiro Nº:5062

Docente: Steven Fortes

Mindelo, 2024

Índice

Objetivo

Desenvolver um sistema capaz de detectar e classificar rosto de aluno em tempo real, utilizando técnica de Transfer Learning para identificar um aluno com exemplo(Anifa) e classificar os demais como “outros”.

Tecnologias Utilizadas:

- OpeanCV: para capturar e processar video da camera.
- Flask: cria uma interface web e reproduz o video em tempo real.
- Haar cascade: para detecção de rostos.
- Transfer Learning: Para reconhecimento facial e classificação (No qual que não ficou implementado com sucesso).

Estrutura do Projecto:

1. Detecção de rosto: utilizei o classificador Haar Cascade para detectar rostos no video em tempo real.

```python
import cv2
faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


You, 3 hours ago | 1 author (You)
class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)

    def __del__(self):
        self.video.release()

    def get_frame(self):
        ret, frame = self.video.read()
        faces = faceDetect.detectMultiScale(frame, 1.3, 5)
        for x, y, w, h in faces:
            x1, y1 = x + w, y + h
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 255), 2)

        # Desenha linhas nos lados do retângulo
            cv2.line(frame, (x, y), (x, y + 30),      You, 3 hours ago • 
                     (255, 0, 255), 6)    # left
            cv2.line(frame, (x, y), (x + 30, y),
                     (255, 0, 255), 6)    # top-left

            cv2.line(frame, (x1, y), (x1 - 30, y),
                     (255, 0, 255), 6)  # right
            cv2.line(frame, (x1, y), (x1, y + 30),
                     (255, 0, 255), 6)  # top-right

            cv2.line(frame, (x, y1), (x, y1 - 30),
                     (255, 0, 255), 6)  # bottom-left
            cv2.line(frame, (x, y1), (x + 30, y1),
                     (255, 0, 255), 6)  # bottom

            cv2.line(frame, (x1, y1), (x1 - 30, y1),
                     (255, 0, 255), 6)  # bottom-right
            cv2.line(frame, (x1, y1), (x1, y1 - 30),
                     (255, 0, 255), 6)  # right-bottom

        ret, jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()
```

Figura 1- codigo referente a camera

2. Falsk como servidor em tempo real:

   Servidor flask para servir o video que captura faces em tempo real.

```python
from flask import Flask, Response, render_template
from camera import Video


app = Flask(__name__)

# Instância da câmera
camera = Video()


@app.route('/')
def index():
    return render_template('index.html')


def gen(camera):
    while True:      You, 3 hours ago • face detect
        frame = camera.get_frame()
        if frame is None:
            break
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame +
               b'\r\n\r\n')


@app.route('/video')
def video():
    return Response(gen(camera),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(debug=True)
```

Figura 2-Servidor Flask

3. Captura de Video
   Foi criada uma classe para encapsular a funcionalidade de captura de video e detencção de rosto.

```
# classificador Haar Cascade para detecção de rostos
faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

while True:
    ret, frame = video.read()
    if not ret:
        break

    # Detecta rostos no frame
    faces = faceDetect.detectMultiScale(frame, 1.3, 5)

    # Desenha retângulos e linhas ao redor dos rostos detectados
    for x, y, w, h in faces:
        x1, y1 = x + w, y + h

        # Desenha o retângulo em volta do rosto
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 255), 2)


        cv2.line(frame, (x, y), (x, y + 30), (255, 0, 255), 6)     # left
        cv2.line(frame, (x, y), (x + 30, y), (255, 0, 255), 6)     # top-

        cv2.line(frame, (x1, y), (x1 - 30, y), (255, 0, 255), 6)  # righ
        cv2.line(frame, (x1, y), (x1, y + 30), (255, 0, 255), 6)  # top-

        cv2.line(frame, (x, y1), (x, y1 - 30), (255, 0, 255), 6)  # bott
        cv2.line(frame, (x, y1), (x + 30, y1), (255, 0, 255), 6)  # bott

        cv2.line(frame, (x1, y1), (x1 - 30, y1),
                 (255, 0, 255), 6)  # bottom-right
        cv2.line(frame, (x1, y1), (x1, y1 - 30),
                 (255, 0, 255), 6)  # right-bottom

    # Exibe o frame com as detecções
    cv2.imshow("Frame", frame)


    k = cv2.waitKey(1)
    if k == ord('q'):
        break

    You, 3 hours ago • Uncommitted changes
video.release()
cv2.destroyAllWindows()
```

Figura 3-Classe vídeo

Desafios Enfrentados:

- Reconhecimento de Aluno especifico: A dificuldade principal foi implementar a parte de reconhecimento facial que diferencia o aluno Anifa de outros alunos. A função face_recognition não foi importada corretamente devido a problemas com a instalação de dependências e compatibilidade com CUDA.

Passos Futuros:

- Resolver problemas de instalação e compatibilidade do face_recognition.
- Implementar Transfer Learning para melhorar a precisão do reconhecimento facial.
- Testar o sistema para garantir que ele possa identificar corretamente o aluno “Anifa” e classificar putros rostos como “outros”.

Referências:

- https://towardsdatascience.com/face-detection-using-face-api-js-and-flask-b641f19228cc
- https://www.youtube.com/watch?v=C_JKIlc_wlU&t=12s
- https://www.youtube.com/watch?v=i_-m1kBTdBI&list=PPSV