Congreso Internacional sobre la Enseñanza y Aplicación de las Matemáticas

Reconocimiento emocional mediante análisis facial con Inteligencia Artificial

Autores:

- Lopez Martinez Hector
- Lopez Sanchez Hugo Yair

Asesora:

- Flores Perez Judith Mayte

correo:

hugoyair.lopez.sanchez@gmail.com

Artículo incluido en la publicación electrónica Memorias del Congreso ISSN 2448-7945 | Mayo 2024

Universidad Nacional Autónoma de México

UNAM CUAUTITLÁN

Departamento de Matemáticas

FESC-INFORMATICA

Objetivo

Desarrollar un sistema de reconocimiento de emociones en imágenes faciales utilizando inteligencia artificial para proporcionar asistencia a personas en el espectro autista, mejorando su comprensión y respuesta ante las expresiones emocionales de los demás.

Introducción

En un mundo donde la tecnología avanza rápidamente, el reconocimiento facial mediante inteligencia artificial se ha convertido en una herramienta poderosa. Este proyecto se centra en aprovechar esta tecnología para abordar una necesidad específica: ayudar a las personas en el espectro autista a interpretar las emociones a través de las expresiones faciales. Esta iniciativa no solo busca mejorar su comprensión emocional, sino también promover una mayor inclusión y conexión en su entorno social.

Metodología y/o desarrollo

En este proyecto de reconocimiento de emociones,se realizó en el lenguaje de programación de **Python** implementando métodos avanzados de reconocimiento facial proporcionados por OpenCV, incluyendo `**EigenFaceRecognizer**`, `**FisherFaceRecognizer**` y `**LBPHFaceRecognizer**`.

Este proyecto consta de cuatro secciones:

1. **Preparación de los datos**
2. **Entrenamiento del modelo**
3. **Lectura del modelo**
4. **Reconocimiento de emociones en tiempo real**

Metodología y/o desarrollo

1. **Preparación de los datos:**
   a. En esta etapa, se recopilan y preparan los datos necesarios para el entrenamiento del modelo de reconocimiento facial. Esto puede incluir la recopilación de imágenes faciales etiquetadas con emociones específicas, como felicidad, tristeza, enojo, sorpresa, etc.
   b. Las imágenes se organizan en una estructura de carpetas donde cada carpeta representa una emoción y contiene las imágenes correspondientes a esa emoción.
   c. Se utiliza OpenCV para cargar las imágenes y convertirlas a escala de grises, ya que los métodos de reconocimiento facial a menudo funcionan mejor con imágenes en escala de grises.

Metodología y/o desarrollo

2. Entrenamiento del modelo:

a) Se utilizan los métodos **EigenFaceRecognizer**, **FisherFaceRecognizer** y **LBPHFaceRecognizer** proporcionados por OpenCV para entrenar el modelo de reconocimiento facial.
b) Durante el entrenamiento, el modelo aprende a reconocer patrones en las imágenes faciales que están asociadas con diferentes emociones.
c) Para cada método de reconocimiento facial, se pasan las imágenes de entrenamiento y las etiquetas correspondientes que indican la emoción asociada con cada imagen.
d) El modelo aprende a mapear las características de las imágenes faciales a las emociones correspondientes durante este proceso de entrenamiento.

Metodología y/o desarrollo

3. Lectura del modelo:

a) Una vez que el modelo ha sido entrenado, se guarda en un archivo XML utilizando el método write() proporcionado por OpenCV.

b) Este archivo XML contiene la información necesaria para que el modelo pueda ser utilizado para reconocer emociones en imágenes faciales nuevas sin necesidad de volver a entrenar el modelo desde cero cada vez que se ejecuta el programa.

Metodología y/o desarrollo

4. Reconocimiento de emociones en tiempo real:

a) Se utiliza la cámara web o se cargan videos o imágenes para detectar caras y reconocer emociones en tiempo real.
b) Se utilizan métodos de detección de caras, como el clasificador de cascada de Haar, proporcionado por OpenCV para identificar las caras en la entrada de video o imágenes.
c) Una vez que se detecta una cara, se extrae y preprocesa la región facial correspondiente para garantizar que tenga el mismo formato que las imágenes utilizadas durante el entrenamiento.
d) Se pasa la región facial preprocesada al modelo de reconocimiento facial para predecir la emoción asociada con esa cara.
e) Dependiendo del método de reconocimiento facial utilizado, se establece un umbral de confianza para la predicción de la emoción. Si la confianza de la predicción supera este umbral, se muestra la emoción predicha en la pantalla junto con la cara detectada. Si no, se muestra "Desconocido".

Desarrollo - capturandoRostros.py

capturandoRostros.py > ...

```python
import cv2
import os
import imutils

emotionName = 'Enojo'
#emotionName = 'Felicidad'
#emotionName = 'Sorpresa'
#emotionName = 'Tristeza'
dataPath = 'C:/Users/HugoLopez2809/Desktop/ReconocimientoEmociones/data'
emotionsPath = dataPath + '/' + emotionName
#print(personPath)
if not os.path.exists(emotionsPath):
  print('Carpeta creada: ', emotionsPath)
  os.makedirs(emotionsPath)
#CAMBIO DE ENTRADA DE DATOS
#cap = cv2.VideoCapture('YairTest1.mp4')
cap=cv2.VideoCapture(0,cv2.CAP_DSHOW)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
count = 300
while True:
  ret, frame = cap.read()
  if ret == False : break
  frame= imutils.resize(frame, width = 640)
  gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
  auxFrame = frame.copy()
  faces = faceClassif.detectMultiScale(gray,1.3,5)
  for (x,y,w,h) in faces:
    cv2.rectangle (frame,(x,y),(x+w,y+h),(0,255,0),2)
    rostro = auxFrame[y:y+h,x:x+w]
    rostro = cv2.resize(rostro,(150,150),interpolation= cv2.INTER_CUBIC)
    cv2.imwrite(emotionsPath+'/rostro_{}.jpg'.format(count),rostro)
```

Desarrollo - capturandoRostros.py

```
33          count = count + 1
34      cv2.imshow('frame',frame)
35      k = cv2.waitKey(1)
36      if k == 27 or count>=400:
37          break
38  cap.release()
39  cv2.destroyAllWindows()
```

Desarrollo - entrenando.py

entrenando.py > ...

```python
import cv2
import os
import numpy as np
import time
def obtenerModelo(method,facesData,labels):
  if method == 'EigenFaces':emotion_recognizer = cv2.face.EigenFaceRecognizer_create()
  if method == 'FisherFaces':emotion_recognizer = cv2.face.FisherFaceRecognizer_create()
  if method == 'LBPHFaces':emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()
  #ENTRENAR EL RECONOCEDOR
  print('Entrenando ('+method+')...')
  inicio= time.time()
  emotion_recognizer.train(facesData, np.array(labels))
  tiemporEntrenamiento=time.time()-inicio
  print('Tiempo de entrenamiento: ('+method+'): ',tiemporEntrenamiento)
  #almacenando modelo de entrenamineto
  emotion_recognizer.write('modelo'+method+'.xml')

dataPath = 'C:/Users/HugoLopez2809/Desktop/ReconocimientoEmociones/data'
emotionList = os.listdir(dataPath)
print('Lista de personas: ',emotionList)
labels = []
facesData= []
label =0
for nameDir in emotionList:
  emotionPath = dataPath+'/'+nameDir
  #print('Leyendo las imagenes')
  for fileName in os.listdir(emotionPath):
    #print('Rostros: ', nameDir +'/'+fileName)
    labels.append(label)
    facesData.append(cv2.imread(emotionPath+'/'+fileName,0))
    image = cv2.imread(emotionPath+'/'+fileName,0)
    #cv2.imshow('image',image)
```

Desarrollo - entrenando.py

```
33          #cv2.waitKey(10)
34        label = label+1
35    obtenerModelo('EigenFaces',facesData,labels)
36    obtenerModelo('FisherFaces',facesData,labels)
37    obtenerModelo('LBPHFaces',facesData,labels)
38    print("Modelo almacenado...")
```

Desarrollo - ReconocimientoEmociones.py

```python
import cv2
import os
import numpy as np

def emotionImage(emotion):
    #emojis
    if emotion == 'Felicidad':image = cv2.imread('emojis/feliz.png')
    if emotion == 'Enojo':image = cv2.imread('emojis/enojo.png')
    if emotion == 'Sorpresa':image = cv2.imread('emojis/sorprendido1.png')
    if emotion == 'Tristeza':image = cv2.imread('emojis/triste.png')
    return image

dataPath = 'C:/Users/HugoLopez2809/Desktop/ReconocimientoEmociones/data'
imagePaths = os.listdir(dataPath)
print('imagePaths = ',imagePaths)

#------------ Métodos usados en el entrenamineto y lectura del modelo ----------
#method = 'EigenFaces'
#method = 'FisherFaces'
method = 'LBPHFaces'
if method == 'EigenFaces':emotion_recognizer = cv2.face.EigenFaceRecognizer_create()
if method == 'FisherFaces':emotion_recognizer = cv2.face.FisherFaceRecognizer_create()
if method == 'LBPHFaces':emotion_recognizer = cv2.face.LBPHFaceRecognizer_create()

#leyendo el modelo
emotion_recognizer.read('modelo'+method+'.xml')
#-------------------------------------------------------------------------------
cap=cv2.VideoCapture(0,cv2.CAP_DSHOW)

#cap = cv2.VideoCapture('Imagenes y Videos de Prueba/YairTest.mp4')
#cap = cv2.VideoCapture('Imagenes y Videos de Prueba/YairTest1.mp4')
#cap = cv2.VideoCapture('Imagenes y Videos de Prueba/Daniel.mp4')
```

Desarrollo - ReconocimientoEmociones.py

```python
#cap = cv2.VideoCapture('Imagenes y Videos de Prueba/Test1.mp4')
#cap = cv2.VideoCapture('Imagenes y Videos de Prueba/Yair.mp4')
faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

while True:
  ret, frame = cap.read()
  if ret == False:break
  gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
  auxFrame =gray.copy()
  nFrame = cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])

  faces=faceClassif.detectMultiScale(gray,1.3,5)
  for (x,y,w,h) in faces:
    rostro = auxFrame[y:y+h,x:x+w]
    rostro = cv2.resize(rostro,(150,150),interpolation = cv2.INTER_CUBIC)
    result = emotion_recognizer.predict(rostro)

    cv2.putText(frame,'{}'.format(result),(x,y-5),1,1.3,(255,255,0),1,cv2.LINE_AA)
    if method == 'EigenFaces':
      #EigenFaces valores de confianza

      if result[1]<5900:
        cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        image = emotionImage(imagePaths[result[0]])
        # Asegúrate de que 'image' y 'frame' tengan las mismas dimensiones
        if image is not None:
          image = cv2.resize(image, (frame.shape[1], frame.shape[0]))
        nFrame = cv2.hconcat([frame,image])
      else:
        cv2.putText(frame,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
```

Desarrollo - ReconocimientoEmociones.py

```
65            nFrame = cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])
66        if method == 'FisherFaces':
67            #fisherFace
68            if result[1]<500:
69                cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
70                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
71                image = emotionImage(imagePaths[result[0]])
72                # Asegúrate de que 'image' y 'frame' tengan las mismas dimensiones
73                if image is not None:
74                    image = cv2.resize(image, (frame.shape[1], frame.shape[0]))
75                nFrame = cv2.hconcat([frame,image])
76            else:
77                cv2.putText(frame,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
78                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
79                nFrame = cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])
80        if method == 'LBPHFaces':
81            #LBPH
82            if result[1]<70:
83                cv2.putText(frame,'{}'.format(imagePaths[result[0]]),(x,y-25),2,1.1,(0,255,0),1,cv2.LINE_AA)
84                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
85                image = emotionImage(imagePaths[result[0]])
86                # Asegúrate de que 'image' y 'frame' tengan las mismas dimensiones
87                if image is not None:
88                    image = cv2.resize(image, (frame.shape[1], frame.shape[0]))
89                nFrame = cv2.hconcat([frame,image])
90            else:
91                cv2.putText(frame,'Desconocido',(x,y-20),2,0.8,(0,0,255),1,cv2.LINE_AA)
92                cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),2)
93                nFrame = cv2.hconcat([frame,np.zeros((480,300,3),dtype=np.uint8)])
94    cv2.imshow('nFrame',nFrame)
95    k= cv2.waitKey(1)
```

Desarrollo - ReconocimientoEmociones.py

```
96      if k == 27:
97          break
98  cap.release()
99  cv2.destroyAllWindows()
```

Metodología y/o desarrollo

En el desarrollo de este proyecto de machine learning se usó un modelo de **Redes Neuronales Convolucionales (CNN)**, Se creó y usó una metodología de un modelo pre-entrenado. Estos datos contienen imágenes de rostros etiquetadas con distintas emociones.

También se usaron archivos para la detección única y precisa de rostros, ya que esto era necesario para no confundir cualquier otra parte del cuerpo y el modelo intentara predecir la emoción de una parte que no sea una cara.

Red Neuronal Convolucional - Emotion_detection.py

emotion_detection.py > ...

```python
import cv2
from keras.models import load_model
from keras.optimizers import Adam
import numpy as np

# Cargar el modelo de reconocimiento de emociones pre-entrenado
model = load_model('model.h5')
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

# Cargar el archivo haarcascade para la detección de rostros
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default (2).xml')

# Diccionario de etiquetas de emociones
emotions = {0: 'Enojado', 1: 'Disgusto', 2: 'Miedo', 3: 'Feliz', 4: 'Triste', 5: 'Sorprendido'

# Capturar video de la cámara
cap = cv2.VideoCapture(0)

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

    # Convertir a escala de grises
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detectar rostros en la imagen
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

Red Neuronal Convolucional - Emotion_detection.py

```
# Para cada rostro detectado
for (x, y, w, h) in faces:
    # Recortar la región del rostro
    face_roi = gray[y:y+h, x:x+w]

    # Redimensionar la imagen para que coincida con el tamaño del modelo (48x48)
    resized = cv2.resize(face_roi, (48, 48))

    # Normalizar la imagen
    normalized = resized / 255.0

    # Redimensionar para que coincida con las dimensiones de entrada del modelo (1, 48, 48,
    reshaped = np.reshape(normalized, (1, 48, 48, 1))

    # Predecir la emoción
    result = model.predict(reshaped)

    # Obtener la etiqueta de la emoción
    emotion_label = emotions[np.argmax(result)]

    # Mostrar la emoción detectada en la imagen
    cv2.putText(frame, emotion_label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12)

    # Dibujar un rectángulo alrededor del rostro detectado
    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

# Mostrar el video en tiempo real
cv2.imshow('Emotion Detection', frame)
```

Red Neuronal Convolucional - Emotion_detection.py

```
59        # Detener la ejecución al presionar la tecla 'q'
60        if cv2.waitKey(1) & 0xFF == ord('q'):
61            break
62
63    # Liberar la captura de la cámara y cerrar las ventanas
64    cap.release()
65    cv2.destroyAllWindows()
66
```

Red Neuronal Convolucional - haarcascade frontal face.xml

haarcascade_frontalface_default (2).xml

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
```

Red Neuronal Convolucional - haarcascade frontal face.xml

haarcascade_frontalface_default (2).xml

```
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <stages>
    <_>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 19 1.5814000740647316e-02</internalNodes>
          <leafValues>
            -4.1960600018501282e-01 1.3576040267944336e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 20 -2.0700000226497650e-02</internalNodes>
          <leafValues>
            1.4590020179748535e+00 -1.9739399850368500e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 21 -1.3760800659656525e-01</internalNodes>
          <leafValues>
            1.1186759471893311e+00 -5.2915501594543457e-01</leafValues></_>
        <_>
          <internalNodes>
            0 -1 22 1.4318999834358692e-02</internalNodes>
          <leafValues>
            -3.5127198696136475e-01 1.1440860033035278e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 23 1.0253000073134899e-02</internalNodes>
          <leafValues>
```

Red Neuronal Convolucional - haarcascade frontal face.xml

haarcascade_frontalface_default (2).xml

```
46    <cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
14721   <features>
18816     <_>
18822     <_>
18823       <rects>
18824         <_>
18825           1 10 18 4 -1.</_>
18826         <_>
18827           7 10 6 4 3.</_></rects></_>
18828     <_>
18829       <rects>
18830         <_>
18831           13 4 8 10 -1.</_>
18832         <_>
18833           17 4 4 5 2.</_>
18834         <_>
18835           13 9 4 5 2.</_></rects></_>
18836     <_>
18837       <rects>
18838         <_>
18839           7 8 9 6 -1.</_>
18840         <_>
18841           10 8 3 6 3.</_></rects></_>
18842     <_>
18843       <rects>
18844         <_>
18845           12 9 9 8 -1.</_>
18846         <_>
18847           15 9 3 8 3.</_></rects></_>
18848     <_>
```

Red Neuronal Convolucional - haarcascade frontal face.xml

```
33293                3 19 9 2 3.</_></rects></_>
33294            <_>
33295              <rects>
33296                <_>
33297                  16 2 3 20 -1.</_>
33298                <_>
33299                  17 2 1 20 3.</_></rects></_>
33300            <_>
33301              <rects>
33302                <_>
33303                  0 13 24 8 -1.</_>
33304                <_>
33305                  0 17 24 4 2.</_></rects></_>
33306            <_>
33307              <rects>
33308                <_>
33309                  9 1 6 22 -1.</_>
33310                <_>
33311                  12 1 3 11 2.</_>
33312                <_>
33313                  9 12 3 11 2.</_></rects></_></features></cascade>
33314    </opencv_storage>
33315
```

Resultados

Finalmente, se mostraron las emociones predichas junto con las caras detectadas en la pantalla, permitiendo una interacción en tiempo real con las emociones expresadas por los sujetos en las imágenes.

Conclusiones

Gracias a este proyecto se aprendió a que la metodología implica la preparación de datos, el entrenamiento del modelo utilizando diferentes métodos de reconocimiento facial proporcionados por OpenCV, la lectura del modelo entrenado y, finalmente, el uso del modelo para detectar y reconocer emociones en tiempo real en imágenes faciales nuevas.

Bibliografía

- Solano.G . (2022, 29 agosto). 🕵️‍♂️ RECONOCIMIENTO FACIAL usando Face Recognition | Python – OpenCV. Recuperaado el 25 de abril de 2024 de https://omes-va.com/face-recognition-python/
- Rosebrock, A. (2023, 8 junio). Face recognition with OpenCV, Python, and deep learning - PyImageSearch. PyImageSearch. Recuperaado el 25 de abril de 2024 de
- https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
- Garcés.C (2021,27 mayo), Reconocer y gestionar las emociones para mejorar la salud mental. Recuperaado el 25 de abril de 2024 de https://uchile.cl/noticias/176402/reconocer-y-gestionar-las-emociones-para-mejorar-la-salud-mental
- Turk, M., & Pentland, A. (1991). Eigenfaces for Recognition. Journal Of Cognitive Neuroscience, 3(1), 71-86. Recuperaado el 25 de abril de 2024 de https://doi.org/10.1162/jocn.1991.3.1.71
- Anggo, M., & Arapu, L. (2018). Face recognition using Fisherface method. Journal Of Physics. Conference Series, 1028, 012119. Recuperaado el 25 de abril de 2024 de https://doi.org/10.1088/1742-6596/1028/1/012119
- Prado, K. S. D. (2018, 19 junio). Face Recognition: Understanding LBPH Algorithm - towards Data Science. Medium. Recuperaado el 25 de abril de 2024 de https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

Gracias por su atención


Departamento de Matemáticas