**UNIVERSIDAD AUTONOMA LATINOAMERICANA**

**INTELIGENCIA ARTIFICIAL**

**31 de octubre de 2023**

**INFORME**

**Alen Yair Camargo Zapata**

**Víctor Alejandro Garcia Betancur**

**INFORME DE APLICACIONN DE MASK DETECTOR**

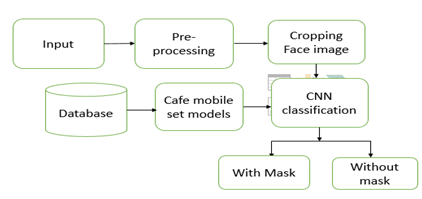
**Contexto:**

El reconocimiento facial es un área prometedora de la visión por ordenador aplicada. Esta técnica se utiliza para reconocer una cara o identificar a una persona automáticamente a partir de imágenes dadas. En nuestras actividades cotidianas, como la comprobación de pasaportes, puertas inteligentes, control de acceso, verificación de votantes, investigación criminal y muchos otros fines, el reconocimiento facial se utiliza ampliamente para autenticar a una persona de forma correcta y automática. El reconocimiento facial ha ganado mucha atención como una tecnología de reconocimiento biométrico única y fiable que la hace más popular que cualquier otra técnica biométrica como la contraseña, el pin, la huella dactilar, etc. Muchos gobiernos de todo el mundo también están interesados en el sistema de reconocimiento facial para proteger lugares públicos como parques, aeropuertos, estaciones de autobús, estaciones de tren, etc. El reconocimiento facial es uno de los problemas de la vida real mejor estudiados. La tecnología de reconocimiento facial ha avanzado mucho

**Explicación del proyecto:**

En este informe, se presenta un análisis del código implementado para la creación de un programa el cual nos permite identificar en tiempo real si una persona esta usando o no el tapabocas.

**Análisis del sistema: detección de mascarillas mediante CNN con OpenCV**



**Codigo:**

import cv2

import numpy as np

from tensorflow.keras.models import load\_model

# Cargar el modelo previamente entrenado

model = load\_model('mask\_detector.model')

# Inicializar el detector de caras

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

# Inicializar la cámara (puedes ajustar el número de cámara si es necesario)

cap = cv2.VideoCapture(0)

while True:

# Capturar un marco de la cámara

ret, frame = cap.read()

# Convertir el marco a escala de grises

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# Detectar caras en el marco

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

for (x, y, w, h) in faces:

# Extraer la región de interés (ROI) que contiene la cara

roi = frame[y:y + h, x:x + w]

# Redimensionar la ROI para que coincida con el tamaño esperado por el modelo

roi = cv2.resize(roi, (224, 224))

roi = np.expand\_dims(roi, axis=0) # Agregar una dimensión de lote

# Realizar una predicción sobre la ROI

result = model.predict(roi)

# Mostrar el resultado en función de la predicción

if result[0][0] > 0.5:

cv2.putText(frame, "Con mascarilla", (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

else:

cv2.putText(frame, "Sin mascarilla", (x, y - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)

cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 3)

# Mostrar el marco con las detecciones

cv2.imshow('Detección de mascarillas', frame)

# Detener el bucle si se presiona la tecla 'q'

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Liberar la cámara y cerrar las ventanas

cap.release()

cv2.destroyAllWindows()

**Importación de librerías:**

Se importan las bibliotecas necesarias, incluyendo OpenCV, NumPy y el módulo ‘’load\_model’’ de TensorFlow/Keras para cargar un modelo de detección de mascarillas previamente entrenado.

import cv2

import numpy as np

from tensorflow.keras.models import load\_model

**Carga de datos:**

Se carga un modelo de detección de mascarillas previamente entrenado desde un archivo llamado 'mask\_detector.model'. Este modelo se utilizará para predecir si las personas en la imagen tienen una mascarilla puesta o no.

model = load\_model('mask\_detector.model')

**Se inicia el detector de caras:**

Se inicializa un detector de caras utilizando un clasificador en cascada de Haar. El archivo XML 'haarcascade\_frontalface\_default.xml' contiene información sobre cómo detectar caras en una imagen.

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

**Para iniciar la cámara:**

Se inicializa la cámara para capturar video en tiempo real. El número 0 indica que se utilizará la cámara predeterminada. Puedes ajustar este número si tienes múltiples cámaras conectadas.

cap = cv2.VideoCapture(0)

**Detección de caras:**

Se utiliza el detector de caras previamente inicializado para detectar caras. Los parámetros scaleFactor, minNeighbors y minSize se utilizan para ajustar la sensibilidad y precisión de la detección de caras.

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

**Resultados Obtenidos y Pruebas:**

**Sin mascarilla:**

Imagen que contiene interior, persona, hombre, diente

Descripción generada automáticamente

**Con mascarilla:**

Pantalla de una computadora

Descripción generada automáticamente

**Conclusiones:**

Esta implementación utiliza un modelo de detección de tapabocas en tiempo real haciendo uso de la cámara, esto con el fin de mostrar si las personas llevan tapabocas o no. El modelo se ejecuta en cada cara detectada, y los resultados se muestran en tiempo real.