TECNM

Tecnológico Nacional de México

Campus Culiacán

Ingeniería en Sistemas computacionales

Inteligencia artificial

09:00 – 10:00

Tarea 2 Unidad 4 – Emociones

Integrantes:

Caro García Jorge Ariel

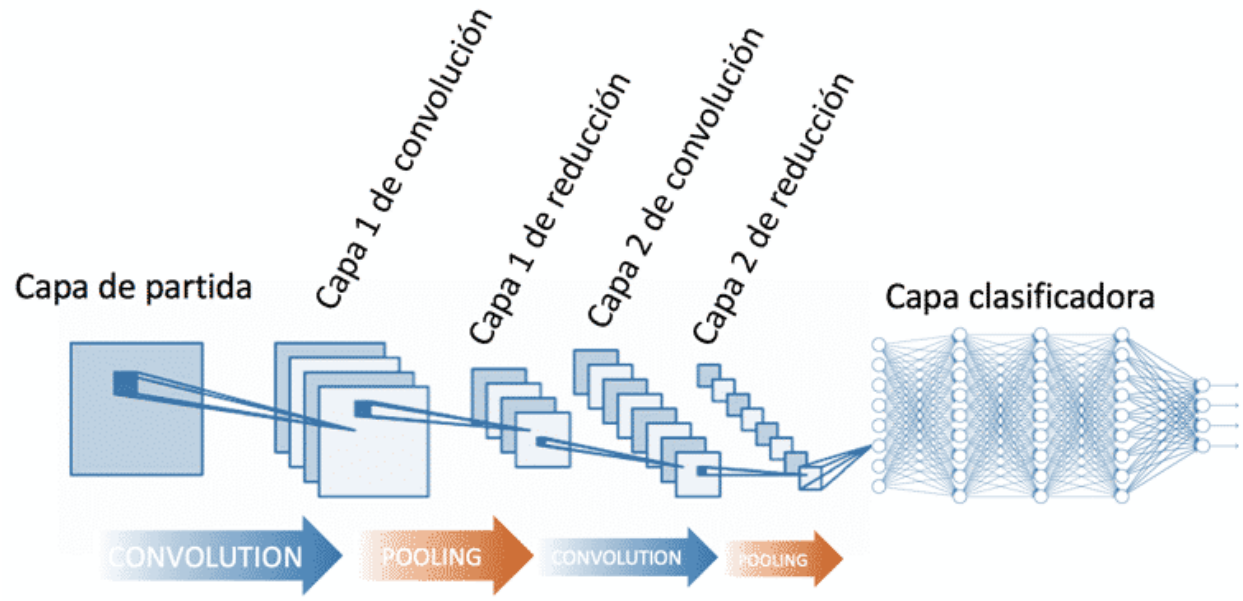
Galván González Sebastián

Docente:

ZURIEL DATHAN MORA FELIX

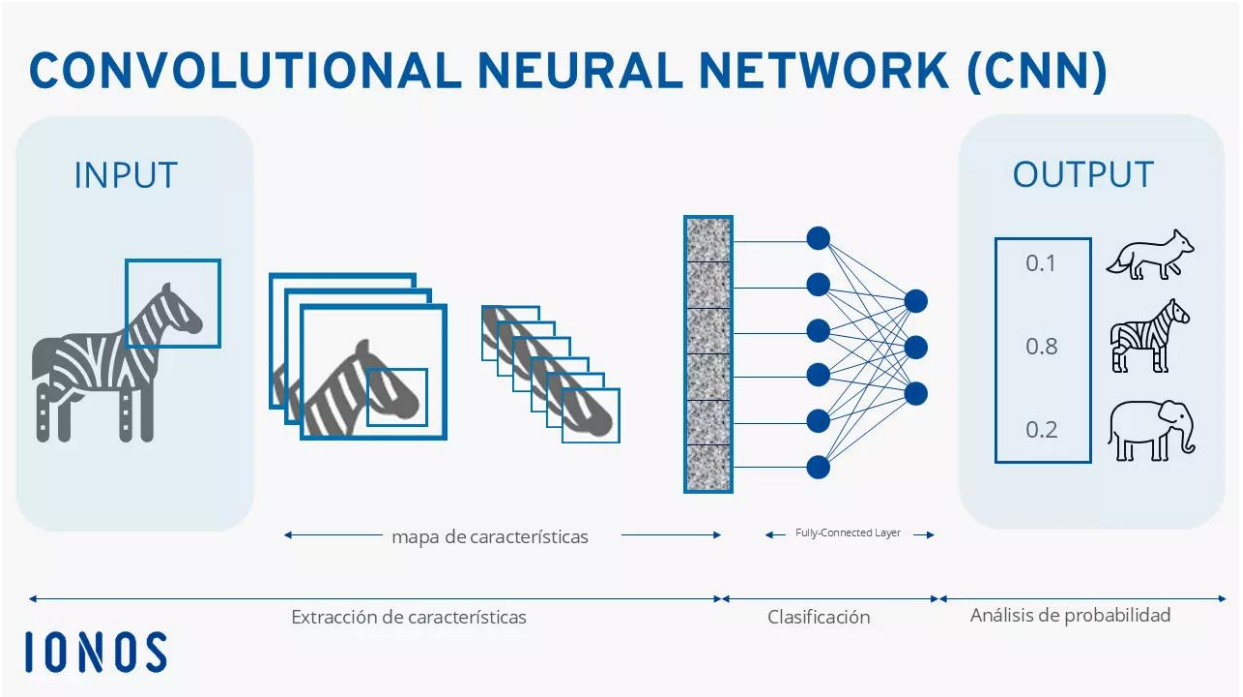
29/05/2025

Arquitectura que utilizamos: Red neuronal convolucional (CNN)



Las CNN designan una subcategoría de redes neuronales y hoy en día son uno de los modelos de clasificación de imágenes considerados como más eficaces.

Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente a partir de datos. Son particularmente útiles para identificar patrones en imágenes con el fin de reconocer objetos, clases y categorías.



Librerias

```
import tensorflow as tf
from keras import layers, models
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split
```

- En cuanto a librerías la única nueva que se agrego es la de scikit-learn, para dividir los datos en entrenamiento y validación.

Configuración y cargar datos

```
# Configura tamaño de imagenes y ruta para el dataset entrenamiento
width, height = 48, 48
ruta_train = 'emociones/ck/CK+48/'

# Cargar imágenes y etiquetas de las emeociones
train_x, train_y = [], []
labels = sorted(os.listdir(ruta_train))
```

- Define el tamaño de las imágenes y la ruta del dataset.
- Obtiene las etiquetas (nombres de las carpetas de emociones).

Preprocesamiento

```
for i in os.listdir(ruta_train):
    for j in os.listdir(os.path.join(ruta_train, i)):
        img_path = os.path.join(ruta_train, i, j)
        img = cv2.imread(img_path)
        if img is None:
            continue
        resized_image = cv2.resize(img, (width, height))
        resized_image = resized_image / 255.0
        train_x.append(resized_image)

        label_vector = np.zeros(len(labels))
        label_vector[labels.index(i)] = 1
        train_y.append(label_vector)
```

- Recorre cada carpeta (emoción) y cada imagen.
- Lee y redimensiona la imagen, normalizándola a valores entre 0 y 1.
- Crea un vector one-hot para la etiqueta correspondiente.
- Guarda la imagen y su etiqueta.

Prepara datos para el modelo

```
x_data = np.array(train_x)
y_data = np.array(train_y)

# divide en entrenamiento y validacion
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.2, random_state=42)
```

- Convierte las listas a arrays de NumPy.
- Divide los datos en entrenamiento 80% y validación 20%.

Entrenamiento del modelo

```
# Modelo CNN
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(width, height, 3)),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(labels), activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=50, validation_data=(x_val, y_val))
model.save('modelo_emociones.keras')
```

- Se deefine una red neuronal convolucional
- Usa softmax en la salida para predecir probabilidades de cada emoción.
- Entrena el modelo y lo guarda.

Carga el modelo entrenado y captura la webcam

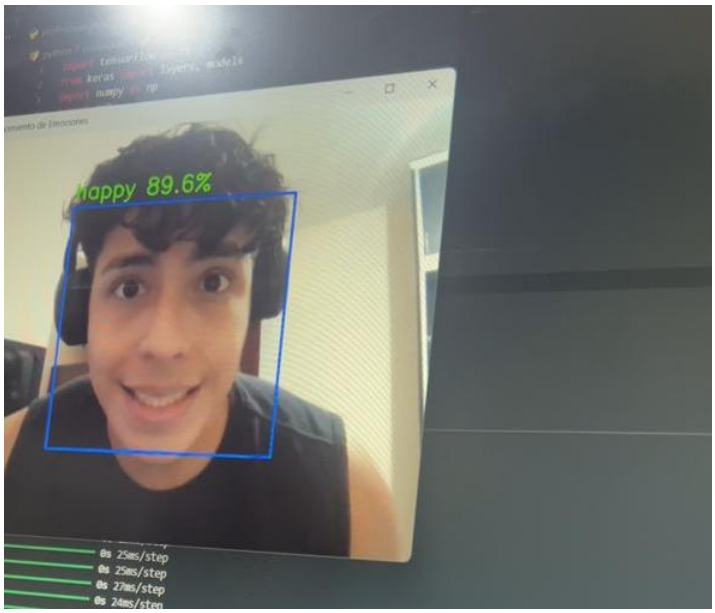
```
# webcam con deteccio de rostro
model = models.load_model('modelo_emociones.keras')
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
```

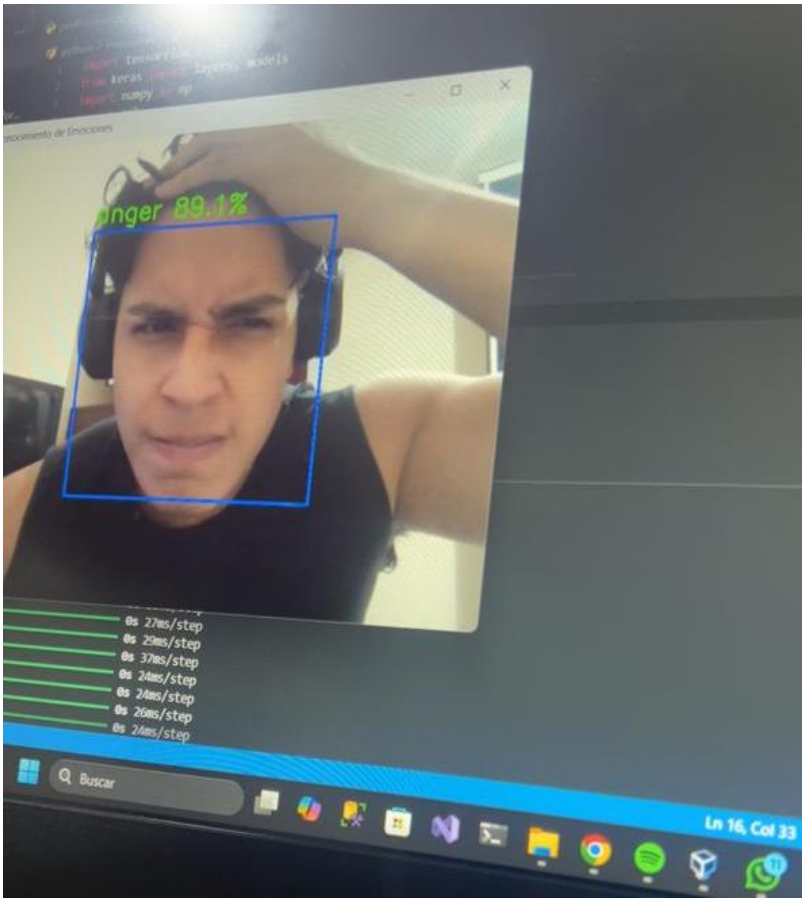
Detectando emociones en tiempo real

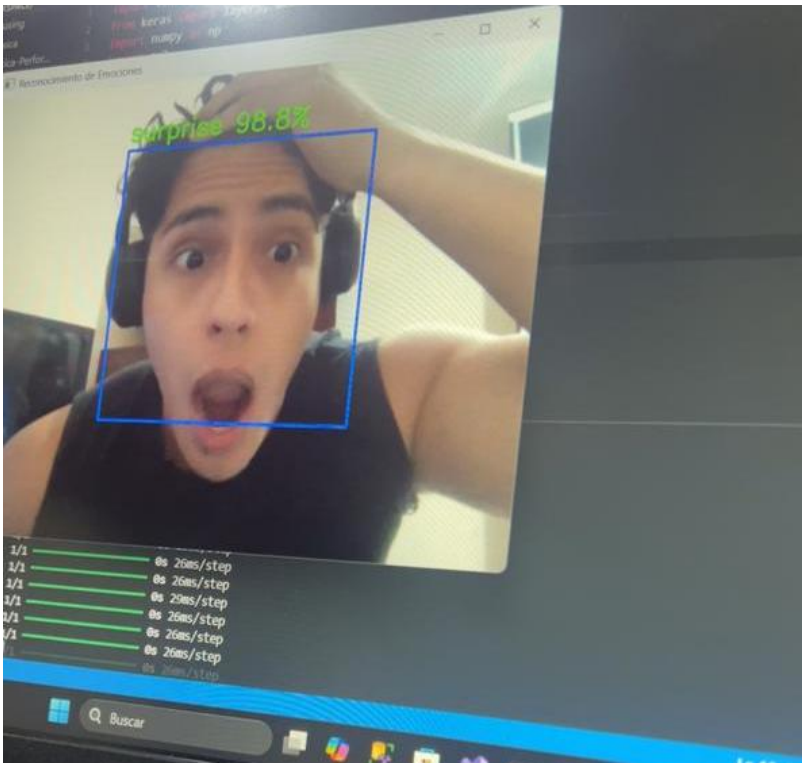
```
print("Presiona 'q' para salir.")
while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        rostro = frame[y:y+h, x:x+w]
        rostro = cv2.resize(rostro, (width, height))
        rostro = rostro / 255.0
        result = model.predict(np.a (variable) result: Any
        emotion = labels[np.argmax(result)]
        porcentaje = max(result) * 100

        cv2.rectangle(frame, (x, y), (x+w, y+h), (255,0,0), 2)
        cv2.putText(frame, f"{emotion} {porcentaje:.1f}%", (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)

    cv2.imshow('Reconocimiento de Emociones', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

- Lee los frames de la webcam y detecta rostros en cda uno
- Redimensiona y normaliza el rostro detectado
- Predice la emoción con el modelo
- Se muestra la emoción encuadrando el rostro y el porcentaje
- Con q salimos del detector

Pruebas




nger 89.1%
0s 27ms/step
0s 29ms/step
0s 37ms/step
0s 24ms/step
0s 24ms/step
0s 26ms/step
0s 24ms/step
Buscar
Ln 16, Col 33


surprise 98.8%
1/1
0s 26ms/step
0s 26ms/step
0s 29ms/step
0s 26ms/step
0s 26ms/step
0s 26ms/step
Buscar