



Maestro: Mauricio Alejandro Cabrera Arellano

Alumno: Nene Miranda José Said 22110319

Materia: Visión Artificial

Tarea: Proyecto

Fecha: 14-06-2025

Proyecto final:

Objetivo general:

Desarrollar un sistema de clasificación de emociones faciales a partir de imágenes usando una red neuronal convolucional (CNN) entrenada con el dataset FER2013.

Objetivos específicos:

- Cargar y procesar el dataset FER2013.
- Construir y entrenar una red neuronal convolucional.
- Evaluar el modelo y mostrar resultados de precisión.
- Documentar el desarrollo del proyecto y su funcionamiento.

Explicación del Proyecto:

Se implementa un sistema que detecta emociones en rostros humanos a partir de imágenes de 48x48 píxeles en escala de grises, usando redes neuronales convolucionales (CNN). Se utilizó el dataset FER2013, disponible en Kaggle, el cual contiene miles de rostros etiquetados con emociones como felicidad, tristeza, enojo, miedo, etc.

El flujo de trabajo incluyó:

- Limpieza de datos.
- Preprocesamiento (normalización y formato).
- Construcción de una red CNN en Keras.
- Entrenamiento con 10 épocas.
- Evaluación y análisis de precisión.

Pruebas y Errores:

Durante el desarrollo se identificaron errores como:

- Filas corruptas en el CSV que causaban errores al hacer reshape.
- Variación brusca en la precisión de validación (resuelto con normalización y dropout).
- Disparidad entre precisión de entrenamiento y validación (se controló con aumento de datos y evaluación por épocas).

6°G_22110319_Proyecto

Los resultados mostraron una tendencia de aprendizaje progresiva, con precisión en validación de alrededor del 35% tras 10 épocas.

Código:

CÓDIGO COMPLETO DE ENTRENAMIENTO CNN CON FER2013

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from sklearn.model_selection import train_test_split

data = pd.read_csv('fer2013.csv')

faces = []

labels = []

for i, row in data.iterrows():

    pixel_sequence = row['pixels']

    pixel_values = [int(pixel) for pixel in pixel_sequence.split()]

    if len(pixel_values) == 48*48:

        face = np.asarray(pixel_values).reshape(48, 48)

        faces.append(face.astype('float32'))

        labels.append(row['emotion'])

faces = np.array(faces)

faces = np.expand_dims(faces, -1)

faces = faces / 255.0

labels = pd.get_dummies(labels).values

X_train, X_test, y_train, y_test = train_test_split(faces, labels, test_size=0.2,
random_state=42)

model = Sequential([
```

6°G_22110319_Proyecto

Conv2D(32, (3,3), activation='relu', input_shape=(48,48,1)),

MaxPooling2D((2,2)),

Conv2D(64, (3,3), activation='relu'),

MaxPooling2D((2,2)),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(7, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=10, batch_size=64, validation_data=(X_test, y_test))

test_loss, test_acc = model.evaluate(X_test, y_test)

print(f"Precisión: {test_acc:.2f}")

Demostración:

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.  
super().__init__(activity_regularizer=activity_regularizer, **kwargs)  
Epoch 1/10  
21/21 6s 182ms/step - accuracy: 0.2287 - loss: 1.8847 - val_accuracy: 0.3477 - val_loss: 1.6179  
Epoch 2/10  
21/21 5s 201ms/step - accuracy: 0.2172 - loss: 1.8536 - val_accuracy: 0.2431 - val_loss: 1.8179  
Epoch 3/10  
21/21 4s 161ms/step - accuracy: 0.2252 - loss: 1.8269 - val_accuracy: 0.2492 - val_loss: 1.7768  
Epoch 4/10  
21/21 5s 167ms/step - accuracy: 0.2726 - loss: 1.7626 - val_accuracy: 0.2389 - val_loss: 1.7672  
Epoch 5/10  
21/21 5s 232ms/step - accuracy: 0.2810 - loss: 1.7638 - val_accuracy: 0.2554 - val_loss: 1.7373  
Epoch 6/10  
21/21 4s 163ms/step - accuracy: 0.3104 - loss: 1.6796 - val_accuracy: 0.2862 - val_loss: 1.7254  
Epoch 7/10  
21/21 5s 163ms/step - accuracy: 0.3454 - loss: 1.6500 - val_accuracy: 0.3231 - val_loss: 1.6997  
Epoch 8/10  
21/21 5s 248ms/step - accuracy: 0.3826 - loss: 1.6883 - val_accuracy: 0.3415 - val_loss: 1.6877  
Epoch 9/10  
21/21 9s 163ms/step - accuracy: 0.4083 - loss: 1.5672 - val_accuracy: 0.3046 - val_loss: 1.6593  
Epoch 10/10  
21/21 6s 221ms/step - accuracy: 0.4391 - loss: 1.5119 - val_accuracy: 0.3169 - val_loss: 1.6579  
11/11 0s 23ms/step - accuracy: 0.3494 - loss: 1.6187
```

