CENTRO DE ENSEÑANZA TECNICA INDUSTRIAL

Detección de Emociones con Visión Artificial y Dataset FER2013

Reporte de proyecto

Victor Alejandro Dominguez Cruz

21310228

6°G

Visión Artificial

15 de junio de 2025

Objetivos del Proyecto

- Desarrollar un sistema capaz de reconocer emociones humanas a partir de expresiones faciales utilizando visión artificial.
- Entrenar un modelo de red neuronal convolucional (CNN) usando el dataset público FER2013.
- Implementar un prototipo en Python que utilice la cámara en tiempo real para detectar rostros y clasificar su emoción.
- Documentar el desarrollo del proyecto, pruebas, errores comunes y soluciones aplicadas.

Explicación del Proyecto

Este proyecto emplea técnicas de visión artificial y aprendizaje profundo para detectar emociones a partir de imágenes faciales captadas por la cámara web. Se utiliza el dataset `Jeneral/fer-2013` disponible en Hugging Face, que contiene miles de imágenes etiquetadas con emociones como: enojo, asco, miedo, felicidad, tristeza, sorpresa y neutral.

El sistema está compuesto por:
- Un script de entrenamiento que entrena una red neuronal convolucional con PyTorch.
- Un script de inferencia que accede a la cámara en tiempo real, detecta rostros con OpenCV y predice la emoción con el modelo entrenado.

Código Implementado

entrenar_emociones_fer2013.py

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
from datasets import load_dataset
from torch.utils.data import DataLoader, Dataset
```

```
from PIL import Image
import io

class EmotionCNN(nn.Module):
    def __init__(self):
        super(EmotionCNN, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, padding=1), nn.ReLU(), nn.MaxPool2d(2),
            nn.Conv2d(16, 32, kernel_size=3, padding=1), nn.ReLU(), nn.MaxPool2d(2)
        )
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32 * 12 * 12, 128), nn.ReLU(),
            nn.Linear(128, 7)
        )

    def forward(self, x):
        return self.fc(self.conv(x))

class FERDataset(Dataset):
    def __init__(self, split="train"):
        dataset = load_dataset("Jeneral/fer-2013", split=split)
        self.images = dataset["img_bytes"]
        self.labels = dataset["labels"]
        self.transform = transforms.Compose([
            transforms.Grayscale(),
```

```
            transforms.Resize((48, 48)),
            transforms.ToTensor()
        ])

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        img = Image.open(io.BytesIO(self.images[idx]))
        img = self.transform(img)
        label = self.labels[idx]
        return img, label

def entrenar_modelo():
    train_dataset = FERDataset("train")
    train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)

    model = EmotionCNN()
    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

    model.train()
    for images, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(images)
        loss = loss_fn(outputs, labels)
```

```
            loss.backward()
            optimizer.step()

    torch.save(model.state_dict(), "modelo_emociones.pt")
    print("Modelo entrenado y guardado como modelo_emociones.pt")
if __name__ == "__main__":
    entrenar_modelo()
```

Detectar emociones

```
import cv2
import torch
import torch.nn as nn
from torchvision import transforms
from PIL import Image

class EmotionCNN(nn.Module):
    def __init__(self):
        super(EmotionCNN, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, padding=1), nn.ReLU(), nn.MaxPool2d(2),
            nn.Conv2d(16, 32, kernel_size=3, padding=1), nn.ReLU(), nn.MaxPool2d(2)
        )
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32 * 12 * 12, 128), nn.ReLU(),
```

```
            nn.Linear(128, 7)
        )

    def forward(self, x):
        return self.fc(self.conv(x))

modelo = EmotionCNN()
modelo.load_state_dict(torch.load("modelo_emociones.pt", map_location=torch.device('cpu')))
modelo.eval()

transform = transforms.Compose([
    transforms.Grayscale(),
    transforms.Resize((48, 48)),
    transforms.ToTensor()
])

clases = ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

print("Presiona ESC 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:
        roi = frame[y:y+h, x:x+w]
        img = Image.fromarray(cv2.cvtColor(roi, cv2.COLOR_BGR2RGB))
        img_tensor = transform(img).unsqueeze(0)
        with torch.no_grad():
            output = modelo(img_tensor)
            pred = torch.argmax(output, dim=1).item()
            emocion = clases[pred]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, emocion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 0, 0), 2)

    cv2.imshow("Reconocimiento de Emociones", frame)
    if cv2.waitKey(1) & 0xFF == 27:
        break

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

Pruebas Realizadas

- Se entrenó el modelo CNN con imágenes del dataset FER2013
- Se verificó el funcionamiento del script de detección con cámara en tiempo real.
- Se probaron múltiples rostros mostrando emociones distintas: alegría, tristeza, sorpresa, etc.

Resultados esperados

- Precisión al detectar emociones claras.
- Latencia aceptable en tiempo real.
- Precisión ante diferentes iluminaciones y expresiones.

Errores y Soluciones

- Error: `DatasetNotFoundError: ashraq/fer2013` no existe.
  Solución: Se utilizó `Jeneral/fer-2013` que es el dataset oficial disponible.
- Error: Las imágenes venían en `img_bytes`, lo que impedía usarlas directamente.
  Solución: Se convirtieron usando `Image.open(io.BytesIO(...))`.
- Error: Malas predicciones con imágenes pequeñas o con mala iluminación.
  Solución: Ajuste del tamaño de entrada y mejora en las condiciones de luz durante la prueba.

Conclusiones

El proyecto logró cumplir con el objetivo de implementar un sistema de visión artificial funcional que detecta emociones humanas en tiempo real. Usando aprendizaje profundo y un dataset abierto, se pudo entrenar un modelo propio sin depender de modelos preentrenados. Este tipo de sistemas puede ser aplicado en interfaces empáticas, salud mental, atención al cliente y educación.

Archivos del Proyecto

- entrenar_emociones_fer2013.py
- detectar_emociones_camara.py
- modelo_emociones.pt

Fuentes

- Hugging Face Datasets: https://huggingface.co/datasets

- FER2013 Dataset: https://huggingface.co/datasets/Jeneral/fer-2013

Enlace GitHub

https://github.com/VictorCet/Inv.-Visi-n-Artificial/upload/main/Proyecto