TECNM

Tecnológico Nacional de México

Campus Culiacán

Ingeniería en Sistemas computacionales

Inteligencia Artificial

09:00 – 10:00

Modelo de detección de emociones

Integrantes:

Beltran Medina Carlos Daniel

Beltrán Ontiveros Karen Valeria

Docente: Mora Félix Zuriel Dathan

29 de mayo de 2025

Detección de Emociones con PyTorch

El presente proyecto consiste en el desarrollo de un modelo de red neuronal convolucional (CNN) para la detección de emociones en rostros humanos a partir de imágenes o video en tiempo real. Este sistema fue implementado utilizando la librería PyTorch y un dataset estructurado en carpetas por clase de emoción. El objetivo principal es clasificar expresiones faciales en siete emociones: enojado, disgustado, aterrado, feliz, neutral, triste y asombrado.

Dataset

Se utilizó un dataset con imágenes organizadas en carpetas para entrenamiento (train/) y prueba (test/), donde cada subcarpeta corresponde a una clase de emoción. Las imágenes fueron convertidas a escala de grises y redimensionadas a 48x48 píxeles.

Preprocesamiento

Para el conjunto de entrenamiento se aplicaron transformaciones como:

- Conversión a escala de grises
- Redimensionamiento
- Aumento de datos (flip horizontal y rotación aleatoria)
- Normalización con media y desviación estándar de 0.5

Arquitectura de la Red Neuronal

Se diseñó una red convolucional sencilla con tres bloques de convolución y capas densas al final:

```
class EmotionCNN(nn.Module):
    def __init__(self):
        super(EmotionCNN, self).__init__()
        self.conv_block = nn.Sequential(
            nn.Conv2d(1, 32, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(32),
nn.MaxPool2d(2),
            nn.Conv2d(32, 64, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(64),
nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(128),
nn.MaxPool2d(2)
        )
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(128 * 6 * 6, 256), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(256, 7)
        )
```

Esta arquitectura permite extraer características espaciales de las imágenes, reducir su dimensionalidad con MaxPool2d, y finalmente clasificar la emoción con una capa densa.

Parámetros de Entrenamiento

- **Épocas:** 30
- **Tamaño de lote (batch size):** 64
- **Tasa de aprendizaje (learning rate):** 0.001
- **Optimización:** Adam
- **Función de pérdida:** CrossEntropyLoss
- **Dispositivo:** GPU (si está disponible), sino CPU

Entrenamiento y Validación

El modelo fue entrenado durante 30 épocas, calculando la pérdida y precisión tanto para el conjunto de entrenamiento como de validación. Se guardó el modelo con mayor precisión en validación. Al final del entrenamiento se generó un reporte de clasificación con classification_report de sklearn.

Reporte de clasificación:

	precision	recall	f1-score	support
angry	0.60	0.49	0.54	958
disgust	0.80	0.37	0.51	111
fear	0.47	0.43	0.45	1024
happy	0.84	0.83	0.84	1774
neutral	0.59	0.63	0.61	1233
sad	0.47	0.56	0.51	1247
surprise	0.77	0.77	0.77	831
accuracy			0.63	7178
macro avg	0.65	0.58	0.60	7178
weighted avg	0.64	0.63	0.63	7178

Prueba con camara en Tiempo Real

Se implementó una interfaz con OpenCV que detecta rostros en vivo mediante un clasificador Haar y predice la emoción correspondiente en pantalla. El modelo cargado es el que obtuvo mejor rendimiento en validación.

Prueba: https://youtu.be/Kg_AlYQMWlU?si=jRpv7Ucm_y7XBeAI

Resultados

El modelo mostró un rendimiento aceptable, logrando diferenciar entre las emociones básicas en tiempo real. La combinación de CNN con procesamiento en GPU permitió un desempeño eficiente.

Conclusión

Este proyecto demostró que es posible desarrollar un sistema funcional de detección de emociones faciales en tiempo real utilizando redes neuronales convolucionales con PyTorch. Gracias al preprocesamiento adecuado y al uso de técnicas como la normalización y el aumento de datos, se logró un buen nivel de precisión en la clasificación. Además, la integración con OpenCV permitió aplicar el modelo en una aplicación práctica con cámara en vivo.