Tecnológico Nacional de México campus Culiacán

Ingeniería en Sistemas Computacionales

Inteligencia Artificial

Entrenamiento del Modelo de machine learning.

Unidad 4

Tarea 2

Darío Corrales Palazuelos

Número de Control:22170616

Proceso

Para pode entrenar el modelo de detección de emociones mediante caras se decidió usar YOLOv8.

YOLOv8 utiliza una arquitectura de red neuronal convolucional (CNN) basada en los principios de YOLO (You Only Look Once), pero con mejoras en cuanto a velocidad, precisión y facilidad de uso respecto a versiones anteriores.

Características principales de la arquitectura de YOLOv8:

1. Backbone (Red troncal):
    - Utiliza una CNN profunda para extraer características de la imagen.
    - Mejoras en la eficiencia computacional respecto a YOLOv5 y anteriores.
2. Neck (Cuello):
    - Emplea PANet (Path Aggregation Network) y FPN (Feature Pyramid Network) para combinar características de diferentes niveles y mejorar la detección de objetos a distintas escalas.
3. Head (Cabeza de detección):
    - Realiza la detección final con múltiples capas convolucionales.
    - Usa anclajes (anchors) optimizados y técnicas de pérdida mejoradas para una mayor precisión.
4. Técnicas avanzadas:
    - Label Assignment: Mejor estrategia de asignación de etiquetas (como Task-Aligned Assigner).
    - Loss Function: Combina pérdidas de clasificación, localización (CIoU, DIOU) y confianza.

Entrenamiento

Una vez instlada la librería de Ultralytics procedemos a configurar un archivo data.yaml donde se encuentran las rutas del entrenamiento así como las de la validación para poder entrenar el modelo.

```yaml
train: /train/images
val: /valid/images
test: /test/images

nc: 6
names: ['anger', 'fear', 'happy', 'neutral', 'sad', 'surprise']

roboflow:
  workspace: dario-corrales
  project: ddddddd-ofrrq-yp6rb
  version: 1
  license: CC BY 4.0
  url: https://universe.roboflow.com/dario-corrales/ddddddd-ofrrq-yp6rb/dataset/1
```

Después se establecen los parámetros del entrenamiento en el código dando a conocer la ubicación del data set y colocando los paramentos como el número de épocas (epochs) que son el número de veces las cuales la red neural observara el data set por completo, el batch que es el numero de imágenes que observara al mismo tiempo y el tamaño de las imágenes a las cuales el programa transforma al tamaño deseado, estos son los parámetros principales al momento de implementar el modelo.

```python
from ultralytics import YOLO
import torch

def train_model():
    model = YOLO("yolov8m.pt")  # O 'yolov8s.pt' para mejor precisión
    results = model.train(
        data="ddddddd.v1i.yolov8\data.yaml",
        epochs=150,  # Aumentar épocas
        batch=16,
        imgsz=640,
        cls=1.5,  # Peso extra para clases minoritarias
        optimizer='AdamW',
        lr0=1e-3,
        patience=30,  # Early stopping si no mejora
        overlap_mask=True,
        name='emociones_v3'
    )

if __name__ == "__main__":
    torch.multiprocessing.freeze_support()  # Crucial para Windows
    train_model()
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      91/100      7.7G     0.7166      0.651       1.05         37        640: 100%|██████████| 21/21 [00:13<00:00,  1.51it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.13s/it]
                   all        114        717      0.561      0.559      0.588      0.415

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      92/100     7.51G     0.7038     0.6271      1.035         46        640: 100%|██████████| 21/21 [00:11<00:00,  1.83it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all        114        717      0.521      0.624      0.589      0.417

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      93/100     7.51G     0.6845     0.5733      1.031         26        640: 100%|██████████| 21/21 [00:11<00:00,  1.84it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:01<00:00,  1.08s/it]
                   all        114        717      0.551      0.569      0.588      0.417

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      94/100     7.51G     0.6715      0.556      1.027         32        640: 100%|██████████| 21/21 [00:11<00:00,  1.85it/s]
```

Resultados del entrenamiento

Se obtuvo los siguiente

Como se puede observar en la Matriz de Confusión Normalizada se ve cómo se confunden las predicciones entre clases, la diagonal principal (de arriba a la derecha a abajo a la izquierda) muestra los aciertos

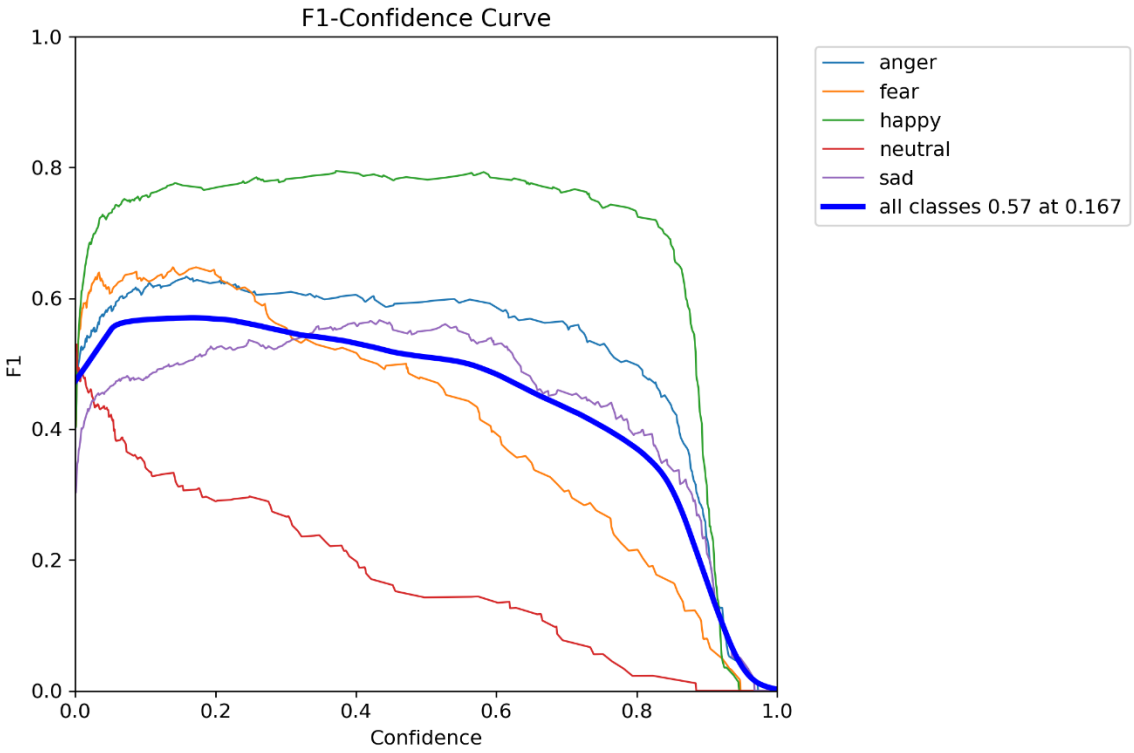
Hallazgos:

- La emoción "happy" tiene la mejor clasificación (76% de aciertos)
- "Anger" se confunde frecuentemente con "sad" (18%) y "background" (16%)
- "Neutral" tiene pobre desempeño (solo 17% de aciertos)
- "Sad" tiene 66% de aciertos pero se confunde con "neutral" (24%)

Curva Precisión-Confianza (P_curve.png) muestra cómo varía la precisión según el umbral de confianza

Hallazgos:

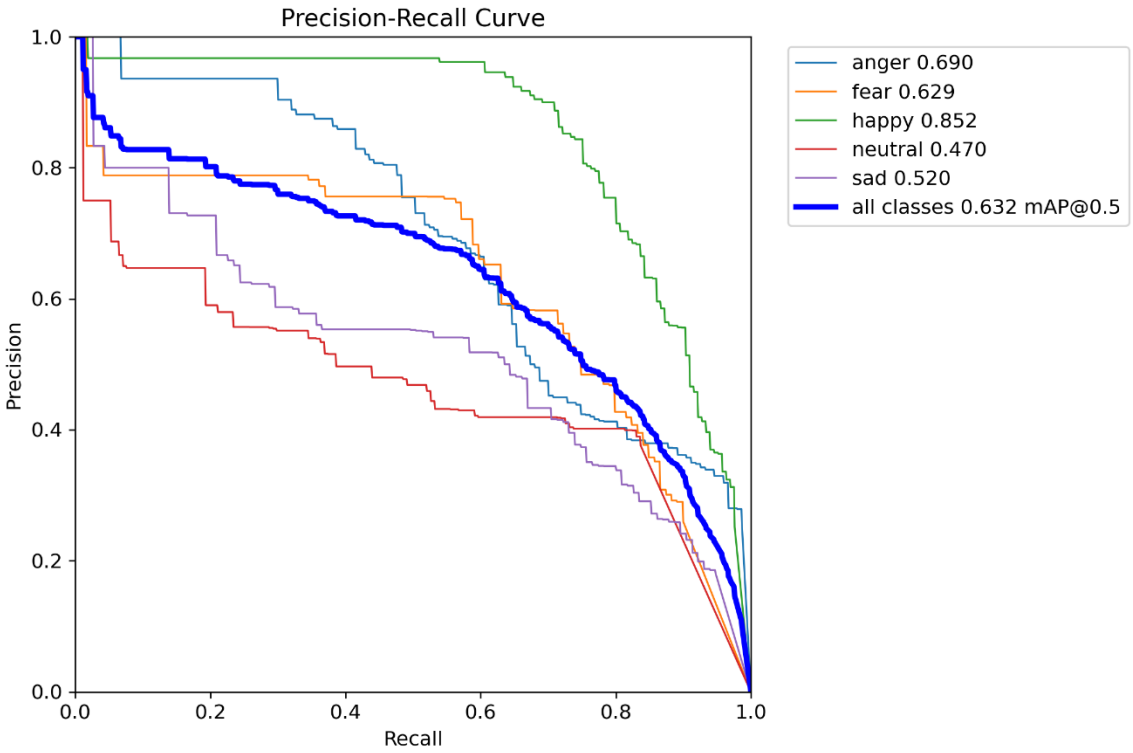
- Alcanza precisión perfecta (1.00) con confianza muy alta (0.998)
- La precisión general parece estar alrededor de 0.4-0.6 para la mayoría de umbrales



Curva Precisión-Recall (PR_curve.png) evalúa el equilibrio entre precisión y exhaustividad

Métricas clave:

- Happy tiene el mejor AP (0.852)
- Neutral tiene el peor desempeño (0.470)
- mAP@0.5 (mean Average Precision) es 0.632, lo que es un valor moderado



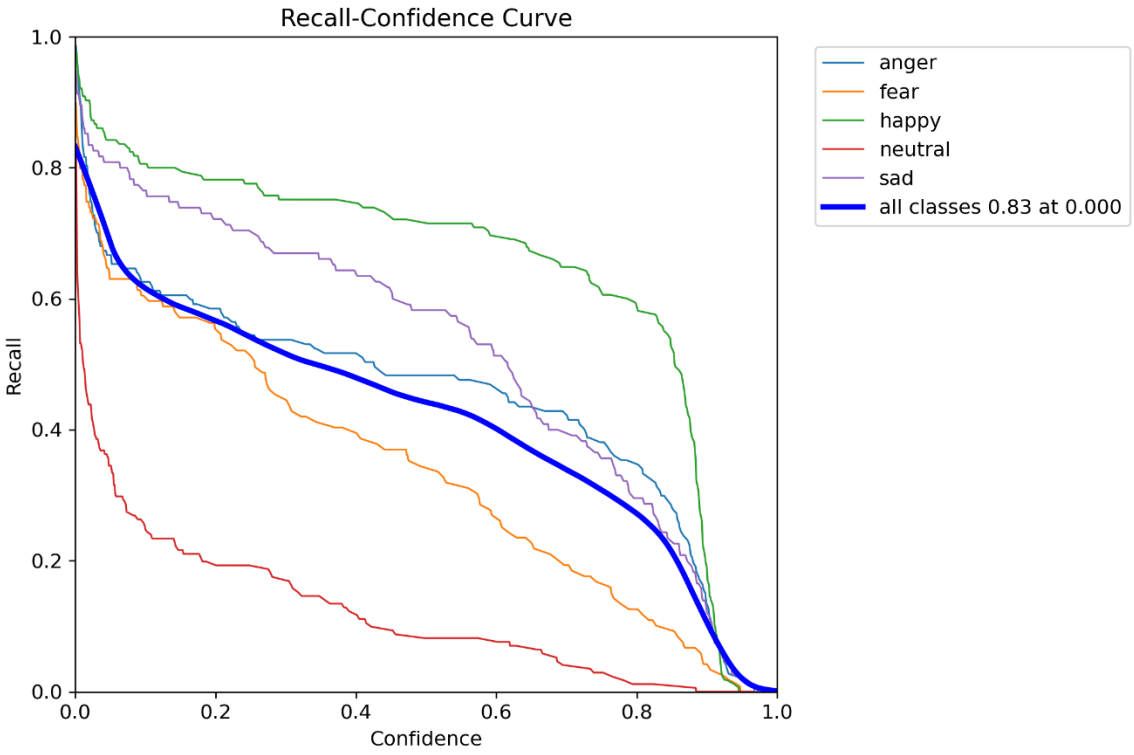
Curva Recall-Confianza (R_curve.png) muestra cómo el recall varía con el umbral de confianza

Hallazgos:

- El recall máximo para todas las clases es 0.83 con confianza 0
- El recall disminuye rápidamente al aumentar la confianza

Problemas identificados:

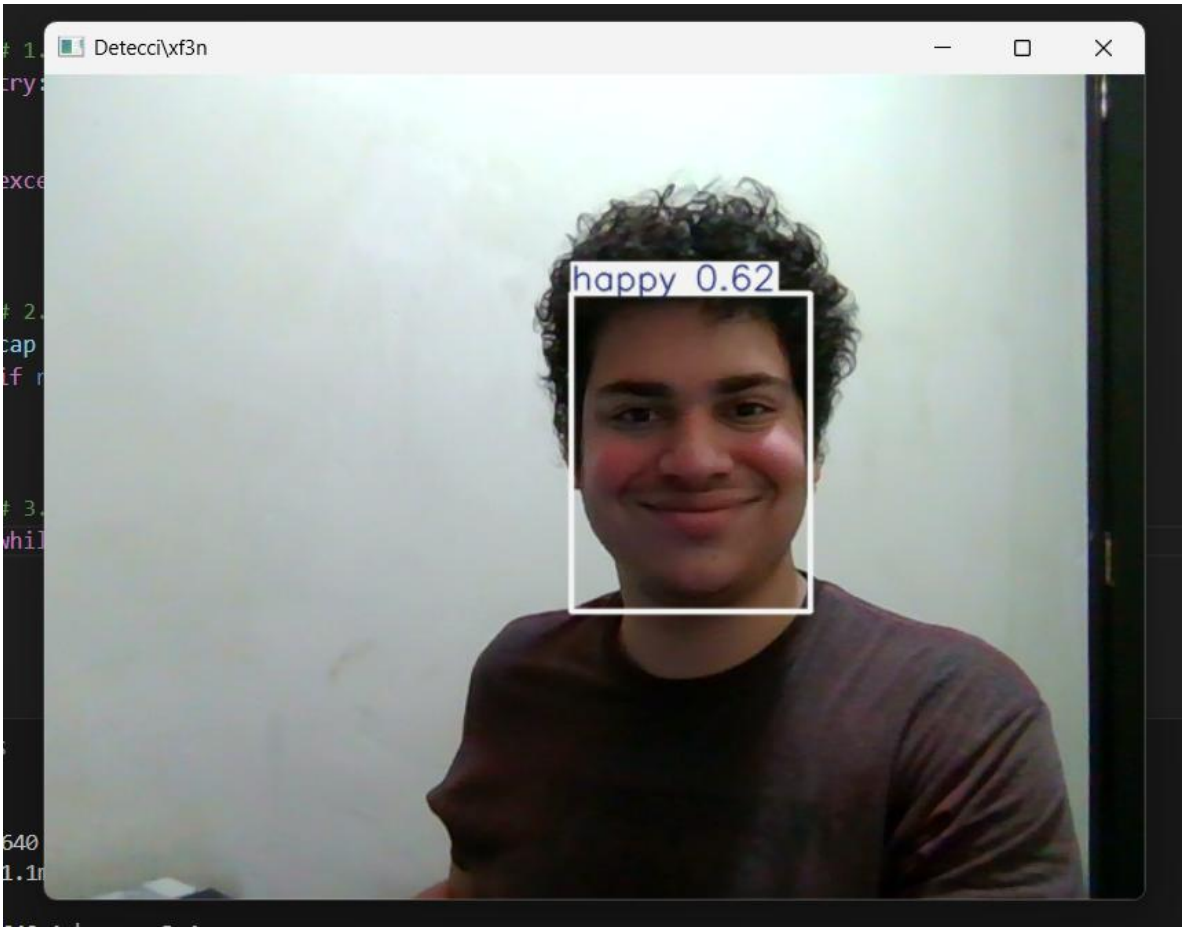
- Clase neutral tiene muy bajo desempeño (17% en matriz de confusión)
- Confusión frecuente entre anger/sad y neutral/background
- Compromiso precisión-recall: Para obtener buena precisión se sacrifica recall y viceversa



Una vez que el modelo se genere este estará listo para probarlo en la Cámara usando la misma librería de Ultralytics importando la librería cv2, para esto se usó el siguiente código.

```python
from ultralytics import YOLO
import cv2

# 1. Cargar modelo con verificación
try:
    model = YOLO('runs/detect/emociones_model3/weights/last.pt')
    print("Clases del modelo:", model.names)
except Exception as e:
    print("Error cargando modelo:", e)
    exit()

# 2. Configurar cámara
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: No se pudo abrir la cámara")
    exit()

# 3. Procesamiento de frames
while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: No se puede leer el frame")
        break

    # 4. Detección con parámetros óptimos
    results = model(frame,
                    imgsz=640,
                    conf=0.5,  # Umbral de confianza
                    iou=0.45,  # Umbral de NMS
                    device='0')  # Usar GPU

    # 5. Mostrar resultados
    if len(results) > 0:
        annotated_frame = results[0].plot()
        cv2.imshow("Detección", annotated_frame)
    else:
        print("No se detectaron objetos")
        cv2.imshow("Detección", frame)

    if cv2.waitKey(1) == ord('q'):
        break
```

Al momento de ejecutarlo se obtuvo lo siguiente



A pesar de funcionar el código y el modelo como se puedo observar en el entrenamiento de este hay una gran confusión entre las emociones de sad y las demás emociones provocando que en muchos casos detecte una emoción errónea.

Al observar esto se generaron otros modelos con data sets y diferentes métricas, a pesar de esto este fue el mejor modelo ya al observar y probar los diversos modelos generados tuvieron una confusión total con la emoción de Sad.