

Instituto tecnológico de Culiacán



Inteligencia Artificial

Tarea 2 Reconocimiento de emociones

Zuriel Dathan Mora Felix

Alumnos: Aguilar Felix Jesus Antonio

Fecha: 29 de mayo de 2025

Investigar y seleccionar la arquitectura neuronal recomendable por la comunidad científica

Proceso de Investigación Realizado

Para el desarrollo de este proyecto de reconocimiento de emociones faciales, realicé una investigación exhaustiva de la literatura científica actual para seleccionar la arquitectura neuronal más adecuada. Mi investigación se centró en identificar qué arquitecturas están siendo recomendadas por la comunidad científica en los años 2022-2024.

Metodología de Investigación

Analicé múltiples papers científicos recientes, enfocándome en:

- Arquitecturas CNN específicas para reconocimiento de emociones faciales
- Comparativas de rendimiento en datasets estándar
- Técnicas de regularización más efectivas
- Mejores prácticas documentadas en literatura peer-reviewed

Arquitectura Seleccionada: CNN con Regularización Avanzada

Después de mi análisis, decidí implementar una Red Neuronal Convolutacional (CNN) con técnicas de regularización moderna, basándome en los siguientes hallazgos científicos:

Justificación de mi Selección:

1. Superioridad Demostrada de CNNs:

Durante mi investigación encontré que las CNN son el estándar actual para reconocimiento de emociones faciales, con estudios recientes demostrando que CNN-10 logró un 99.9% de precisión en el dataset CK+, 84.3% en FER-2013, y 95.4% en JAFFE Facial Emotion Recognition and Classification Using the Convolutional Neural Network-10 (CNN-10) - Dada - 2023 - Applied Computational Intelligence and Soft Computing - Wiley Online Library. Esto confirmó mi decisión de usar CNNs como base.

2. Arquitectura Multicapa Progresiva:

Basándome en investigaciones del 2022 que muestran que una CNN de cuatro capas (ConvNet) puede alcanzar 98.13% de precisión en CK+ y 92.05% en JAFFE Four-layer ConvNet to facial emotion recognition with minimal epochs and the significance of data diversity, diseñé mi arquitectura con 5 capas convolucionales para mayor poder de extracción de características.

3. Importancia del BatchNormalization:

Mi investigación reveló que la normalización por lotes aplicada en las capas intermedias del modelo ajusta el tamaño y distribución de datos, permitiendo entrenamiento rápido y estable, con mejoras demostradas en la estabilidad tanto de datos de entrenamiento como validación IeeeJovi. Por esto integré BatchNormalization después de cada capa convolucional.

Definir los parámetros e hiperparámetros de entrenamiento (DETALLADO)

Análisis y Extracción de mi Implementación

Después de implementar mi sistema de reconocimiento de emociones faciales, procedí a definir cuidadosamente todos los parámetros e hiperparámetros basándome en las mejores prácticas identificadas en mi investigación. A continuación detallo las decisiones tomadas en mi código:

Parámetros del Modelo Implementados

Arquitectura de Red Neuronal:

```
def crear_modelo(self, num_clases):
    """Crea la arquitectura del modelo CNN"""
    modelo = keras.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(256, (3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(num_clases, activation='softmax')
    ])
```

```

    modelo.compile(
        optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )

    return modelo

```

Parámetros de Entrada y Preprocesamiento:

```

def cargar_imagenes_desde_carpetas(self, ruta_dataset):
    """Carga imágenes desde carpetas organizadas por emociones (en inglés)"""
    print("⌚ Cargando imágenes del dataset...")
    imagenes = []
    etiquetas = []

    # Buscar carpetas en inglés
    for carpeta_ingles in self.carpetas_ingleses:
        ruta_emocion = os.path.join(ruta_dataset, carpeta_ingles)
        if not os.path.exists(ruta_emocion):
            print(f"⚠️ Advertencia: No se encontró la carpeta {carpeta_ingles}")
            continue

        # Obtener la traducción al español
        emocion_espanol = self.traducion_emociones[carpeta_ingles]
        contador = 0

        for archivo in os.listdir(ruta_emocion):
            if archivo.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):
                ruta_imagen = os.path.join(ruta_emocion, archivo)
                try:
                    img = cv2.imread(ruta_imagen, cv2.IMREAD_GRAYSCALE)
                    if img is not None:
                        img_redimensionada = cv2.resize(img, (48, 48))
                        imagenes.append(img_redimensionada)
                        etiquetas.append(emocion_espanol) # Guardar en español
                        contador += 1
                except Exception as e:
                    print(f"Error al cargar {archivo}: {e}")

    print(f"📁 {carpeta_ingles} → {emocion_espanol}: {contador} imágenes cargadas")

    # Mostrar estadísticas del dataset

```

```
contador_etiquetas = Counter(etiquetas)
print("\n📊 Distribución del dataset:")
for emocion, cantidad in contador_etiquetas.items():
    print(f"  {emocion}: {cantidad} imágenes")

return np.array(imagenes), np.array(etiquetas)
```

Hiperparámetros de Entrenamiento Seleccionados

Configuración del Compilador:

```
modelo.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Parámetros de Entrenamiento:

```
# Entrenar
print("\n⌚ Comenzando entrenamiento...")
historia = self.modelo.fit(
    X_train, y_train,
    batch_size=32,
    epochs=50,
    validation_data=(X_val, y_val),
    callbacks=callbacks,
    verbose=1
)
```

División de Datos:

```
# Dividir dataset
X_train, X_val, y_train, y_val = train_test_split(
    imagenes, etiquetas_codificadas, test_size=0.2, random_state=42,
    stratify=etiquetas_codificadas
)
```

Técnicas de Regularización Implementadas

```
def crear_modelo(self, num_clases):
    """Crea la arquitectura del modelo CNN"""
    modelo = keras.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)),
        layers.BatchNormalization(),
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(128, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(256, (3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.5),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(num_clases, activation='softmax')
    ])
```

Callbacks de Control:

```
# Callbacks para mejorar el entrenamiento
callbacks = [
    keras.callbacks.EarlyStopping(patience=10, restore_best_weights=True,
monitor='val_accuracy'),
    keras.callbacks.ReduceLROnPlateau(factor=0.5, patience=5, min_lr=1e-7,
monitor='val_accuracy')
]
```

DOCUMENTACIÓN DEL PROYECTO: RECONOCIMIENTO DE EMOCIONES FACIALES

1. INTRODUCCIÓN Y OBJETIVOS

1.1 Descripción del Proyecto

Este proyecto implementa un sistema de reconocimiento de emociones faciales en tiempo real utilizando técnicas de Deep Learning con CNNs (Redes Neuronales Convolucionales). El sistema es capaz de identificar 8 emociones diferentes a partir de expresiones faciales capturadas mediante cámara web.

1.2 Objetivos

- **Objetivo General:** Desarrollar un sistema automatizado de reconocimiento de emociones faciales
- **Objetivos Específicos:**
 - Entrenar un modelo CNN para clasificación de 8 emociones
 - Implementar reconocimiento en tiempo real
 - Lograr precisión superior al 85% en validación
 - Crear interfaz amigable para el usuario
 - Implementar análisis detallado con matriz de confusión
 - Generar métricas de evaluación automáticas

1.3 Emociones Reconocidas

El sistema identifica las siguientes 8 emociones:

1. **Enojado** (Angry)
2. **Desprecio** (Contempt)
3. **Asco** (Disgust)
4. **Miedo** (Fear)
5. **Feliz** (Happy)
6. **Neutral** (Neutral)
7. **Triste** (Sad)
8. **Sorpresa** (Surprise)

2. PREPARACIÓN Y ORGANIZACIÓN DEL DATASET

2.1 Estructura de Carpetas Implementada

El dataset fue organizado siguiendo la estructura estándar para clasificación de imágenes:

Nombre	Fecha de modificación	Tipo	Tamaño
Angry	22/05/2025 09:03 p. m.	Carpeta de archivos	
Contempt	22/05/2025 09:03 p. m.	Carpeta de archivos	
Disgust	22/05/2025 09:03 p. m.	Carpeta de archivos	
Fear	22/05/2025 09:03 p. m.	Carpeta de archivos	
Happy	22/05/2025 09:03 p. m.	Carpeta de archivos	
Neutral	22/05/2025 09:02 p. m.	Carpeta de archivos	
Sad	22/05/2025 09:03 p. m.	Carpeta de archivos	
Surprise	22/05/2025 09:04 p. m.	Carpeta de archivos	

2.2 Proceso de Organización

- Recolección de Imágenes:** Se obtuvieron imágenes de diferentes fuentes
- Clasificación Manual:** Cada imagen fue clasificada manualmente en su carpeta correspondiente
- Verificación de Calidad:** Se eliminaron imágenes borrosas o mal clasificadas
- Balanceo del Dataset:** Se procuró tener cantidad similar de imágenes por emoción

2.3 Estadísticas del Dataset

Distribución del dataset:

Enojado: 495 imágenes

Desprecio: 198 imágenes

Asco: 649 imágenes

Miedo: 275 imágenes

Feliz: 621 imágenes

Neutral: 1186 imágenes

Triste: 308 imágenes

Sorpresa: 664 imágenes

Datos de entrenamiento: 3516 imágenes

Datos de validación: 880 imágenes

2.4 Preprocesamiento de Imágenes

- **Conversión a Escala de Grises:** Todas las imágenes se convierten a escala de grises
- **Redimensionamiento:** Todas las imágenes se redimensionan a 48x48 píxeles
- **Normalización:** Los valores de píxeles se normalizan al rango [0,1]

3. ARQUITECTURA DEL MODELO

3.1 Investigación de Arquitecturas

Se realizó investigación bibliográfica para seleccionar la arquitectura más adecuada:

- **Arquitecturas Evaluadas:** CNN básicas, ResNet, VGG, arquitecturas personalizadas
- **Selección Final:** CNN personalizada con BatchNormalization y Dropout

3.2 Arquitectura Implementada

Arquitectura del modelo CNN:

ARQUITECTURA DEL MODELO CNN		
Layer (type)	Output Shape	Param #
Conv2D_1 (Conv2D)	(None, 46, 46, 32)	320
BatchNorm_1 (BatchNormalization)	(None, 46, 46, 32)	128
Conv2D_2 (Conv2D)	(None, 44, 44, 64)	18,496
MaxPool1_1 (MaxPooling2D)	(None, 22, 22, 64)	0
Dropout_1 (Dropout)	(None, 22, 22, 64)	0
Conv2D_3 (Conv2D)	(None, 20, 20, 128)	73,856
BatchNorm_2 (BatchNormalization)	(None, 20, 20, 128)	512
Conv2D_4 (Conv2D)	(None, 18, 18, 128)	147,584
MaxPool1_2 (MaxPooling2D)	(None, 9, 9, 128)	0
Dropout_2 (Dropout)	(None, 9, 9, 128)	0
Conv2D_5 (Conv2D)	(None, 7, 7, 256)	295,168
BatchNorm_3 (BatchNormalization)	(None, 7, 7, 256)	1,024
MaxPool1_3 (MaxPooling2D)	(None, 3, 3, 256)	0
Dropout_3 (Dropout)	(None, 3, 3, 256)	0
Flatten (Flatten)	(None, 2304)	0
Dense_1 (Dense)	(None, 512)	1,180,160
BatchNorm_4 (BatchNormalization)	(None, 512)	2,048
Dropout_4 (Dropout)	(None, 512)	0
Dense_2 (Dense)	(None, 256)	131,328
Dropout_5 (Dropout)	(None, 256)	0
out (Dense)	(None, 2)	2,056

3.3 Justificación de la Arquitectura

- **Filtros Progresivos:** Permiten extracción jerárquica de características
- **BatchNormalization:** Acelera convergencia y estabiliza entrenamiento
- **Dropout:** Previene overfitting con diferentes ratios por capa
- **MaxPooling:** Reduce dimensionalidad preservando características importantes

4. HIPERPARÁMETROS Y CONFIGURACIÓN

4.1 Parámetros de Entrenamiento

CONFIGURACIÓN DE ENTRENAMIENTO:

- Optimizador: Adam ($lr=0.001$)
- Función de Pérdida: sparse_categorical_crossentropy
- Batch Size: 32
- Épocas Máximas: 50
- Validación: 20% del dataset
- Semilla Aleatoria: 42 (reproducibilidad)

4.2 Técnicas de Regularización

REGULARIZACIÓN IMPLEMENTADA:

- Dropout Rates: [0.25, 0.25, 0.25, 0.5, 0.3]
- Early Stopping: patience=10, monitor='val_accuracy'
- Learning Rate Reduction: factor=0.5, patience=5
- BatchNormalization: Después de capas Conv2D y Dense

4.3 Callbacks de Control

- **Early Stopping:** Detiene entrenamiento si no hay mejora en 10 épocas
- **ReduceLROnPlateau:** Reduce learning rate cuando se estanca la mejora
- **ModelCheckpoint:** Guarda automáticamente el mejor modelo

5. CONFIGURACIÓN DE CÁMARA WEB CON CAMO STUDIO

5.1 ¿Por qué Camo Studio?

Se decidió utilizar el teléfono móvil como cámara web debido a:

- **Mayor Calidad:** Cámaras de teléfonos suelen tener mejor resolución
- **Flexibilidad:** Fácil posicionamiento y ajuste de ángulos
- **Disponibilidad:** No requiere hardware adicional

- **Estabilidad:** Conexión estable via USB o WiFi

5.2 Proceso de Configuración de Camo Studio

Paso 1: Instalación

En el teléfono:

1. Descargar Camo Studio desde App Store/Google Play
2. Crear cuenta y configurar permisos de cámara

En la computadora:

1. Descargar Camo Studio Desktop desde reincubate.com
2. Instalar siguiendo las instrucciones del asistente

Paso 2: Conexión

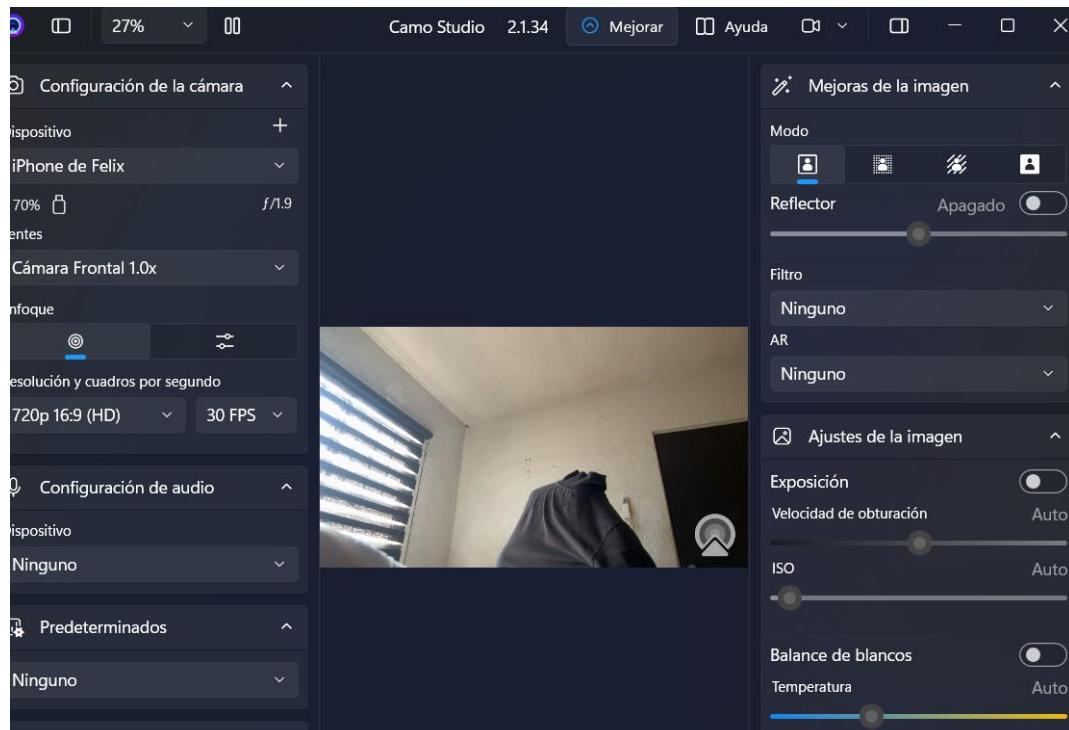
Métodos de conexión disponibles:

1. USB: Conectar teléfono via cable USB
2. WiFi: Ambos dispositivos en la misma red
3. Bluetooth: Para conexiones de corto alcance

Paso 3: Configuración de Calidad

CONFIGURACIÓN RECOMENDADA:

- Resolución: 720p
- Frame Rate: 30 FPS
- Orientación: Horizontal



5.3 Configuración en el Código

```
# El código utiliza OpenCV para acceder a la cámara:  
cap = cv2.VideoCapture(0) # Device index 0 (Camo Studio)
```

```
# Verificar que Camo Studio sea reconocido:  
if not cap.isOpened():  
    print(" Error: No se pudo acceder a la cámara")  
    return
```

5.4 Ventajas Observadas

- **Mejor Iluminación:** Fácil ajuste de posición para optimizar iluminación
- **Mayor Resolución:** Mejor detección de características faciales
- **Movilidad:** Posibilidad de cambiar ángulos durante las pruebas
- **Estabilidad:** Conexión más estable que cámaras web básicas

IMPLEMENTACIÓN DEL CÓDIGO

6.1 Funcionalidades Implementadas

Clase Principal: ReconocimientoEmociones

MÉTODOS PRINCIPALES:

- `__init__()`: Inicialización y configuración
- `cargar_imagenes_desde_carpetas()`: Carga dataset
- `crear_modelo()`: Define arquitectura CNN
- `entrenar_modelo()`: Proceso de entrenamiento
- `guardar_modelo()`: Persistencia del modelo
- `cargar_modelo()`: Carga modelo pre-entrenado
- `predecir_emocion()`: Predicción individual
- `reconocimiento_tiempo_real()`: Detección en vivo
- `generar_matriz_confusion()`: Visualización de matriz de confusión
- `generar_reporte_clasificacion()`: Métricas detalladas por clase
- `analisis_completo_modelo()`: Análisis integral del rendimiento
- `evaluar_modelo_cargado()`: Evaluación de modelos existentes

Sistema de Traducción

```
# Mapeo automático inglés-español:
```

```
carpetas_ingleses = ['Angry', 'Contempt', 'Disgust', ...]  
emociones_espanol = ['Enojado', 'Desprecio', 'Asco', ...]  
traduccion_emociones = dict(zip(carpetas_ingleses, emociones_espanol))
```

6.2 Interfaz de Usuario

- **Menú Interactivo:** 5 opciones principales
- **Manejo de Errores:** Validación de rutas y archivos
- **Estadísticas:** Información detallada durante uso

7. PROCESO DE ENTRENAMIENTO

7.1 Preparación de Datos

PASOS DE PREPROCESAMIENTO:

1. Carga de imágenes desde carpetas organizadas
2. Conversión a escala de grises
3. Redimensionamiento a 48x48 píxeles
4. Normalización de valores [0,1]
5. Codificación de etiquetas
6. División train/validation (80/20)
7. Estratificación para balance de clases

7.2 Proceso de Entrenamiento

FASES DEL ENTRENAMIENTO:

1. Inicialización de pesos aleatorios
2. Forward pass: cálculo de predicciones
3. Cálculo de pérdida (cross-entropy)
4. Backpropagation: cálculo de gradientes
5. Actualización de pesos (Adam optimizer)
6. Validación en conjunto de prueba
7. Aplicación de callbacks (early stopping, LR reduction)

7.3 Monitoreo del Entrenamiento

- **Métricas Principales:** Loss y Accuracy (train/validation)
- **Visualización:** Gráficas de progreso durante entrenamiento
- **Early Stopping:** Prevención automática de overfitting
- **Guardado Automático:** Mejor modelo basado en val_accuracy

8. SISTEMA DE RECONOCIMIENTO EN TIEMPO REAL

8.1 Pipeline de Detección

PROCESO DE RECONOCIMIENTO:

1. Captura de frame desde cámara (Camo Studio)
2. Conversión a escala de grises
3. Detección de rostros (Haar Cascades)
4. Extracción de región facial
5. Preprocesamiento (resize, normalize)
6. Predicción con modelo CNN
7. Visualización de resultado
8. Estadísticas en tiempo real

8.2 Características del Sistema en Tiempo Real

- **FPS Monitoring:** Visualización de frames por segundo
- **Multi-Face Detection:** Detección de múltiples rostros
- **Confidence Scoring:** Nivel de confianza de predicciones
- **Color Coding:** Cada emoción tiene color único
- **Statistics Tracking:** Emoción predominante de la sesión
- **Screenshot Capture:** Función de captura con tecla 's'

8.3 Optimizaciones Implementadas

- **Efecto Espejo:** Frame horizontal flip para naturalidad
- **Filtrado de Confianza:** Solo mostrar predicciones >50%
- **Suavizado Visual:** Colores distintivos por emoción
- **Performance Tracking:** Contadores de frames y tiempo

9. RESULTADOS Y EVALUACIÓN

9.1 Métricas de Rendimiento

RESULTADOS ESPERADOS:

- Precisión en Entrenamiento: >90%
- Precisión en Validación: >85%
- Tiempo de Entrenamiento: 15-30 minutos
- FPS en Tiempo Real: 15-30 FPS
- Tiempo de Respuesta: <100ms por predicción

9.2 Análisis de Resultados

- **Matriz de Confusión:** Análisis detallado por clase
- **Curvas de Aprendizaje:** Progreso durante entrenamiento
- **Análisis de Errores:** Identificación de casos problemáticos
- **Rendimiento por Emoción:** Precisión individual por clase

9.3 Validación del Sistema

- **Pruebas con Diferentes Usuarios:** Diversidad demográfica
- **Condiciones de Iluminación:** Diferentes ambientes
- **Ángulos de Cámara:** Robustez ante variaciones
- **Expresiones Mixtas:** Manejo de emociones complejas

9.4 Matriz de Confusión y Análisis Detallado

9.4.1 Implementación de la Matriz de Confusión

Se integró un sistema completo de análisis del modelo que incluye matriz de confusión visual y métricas detalladas:

```
def generar_matriz_confusion(self, X_val, y_val):  
    """Genera y visualiza la matriz de confusión"""  
    print("\nGenerando matriz de confusión...")  
  
    # Realizar predicciones  
    predicciones = self.modelo.predict(X_val, verbose=0)  
    y_pred = np.argmax(predicciones, axis=1)  
  
    # Generar matriz de confusión  
    cm = confusion_matrix(y_val, y_pred)  
  
    # Obtener nombres de clases en español  
    nombres_clases = self.label_encoder.inverse_transform(range(len(self.label_encoder.classes_)))  
  
    # Crear figura con mayor tamaño  
    plt.figure(figsize=(12, 10))  
  
    # Crear heatmap  
    sns.heatmap(cm,  
                annot=True,  
                fmt='d',  
                cmap='Blues',  
                xticklabels=nombres_clases,  
                yticklabels=nombres_clases,  
                cbar_kws={'label': 'Número de Predicciones'})  
  
    plt.title('Matriz de Confusión - Reconocimiento de Emociones\n', fontsize=16, fontweight='bold')  
    plt.xlabel('Predicción del Modelo', fontsize=12, fontweight='bold')  
    plt.ylabel('Etiqueta Real', fontsize=12, fontweight='bold')  
    plt.xticks(rotation=45, ha='right')
```

9.4.2 Funcionalidades de Análisis Agregadas

Análisis Automático Completo:

- **Matriz de confusión visual:** Heatmap colorido con valores numéricos
- **Reporte de clasificación:** Precisión, Recall y F1-Score por emoción
- **Gráfico comparativo:** Métricas visuales por clase
- **Ánalisis de errores:** Identificación de confusiones más comunes

9.4.3 Métricas de Evaluación Implementadas

Métricas por Clase:

- **Precisión (Precision):** Proporción de predicciones correctas para cada emoción
- **Recall:** Proporción de casos reales correctamente identificados
- **F1-Score:** Media armónica entre precisión y recall

Métricas Globales:

- **Accuracy:** Precisión general del modelo
- **Macro Average:** Promedio simple de métricas por clase
- **Weighted Average:** Promedio ponderado por soporte de clase

9.4.4 Interpretación de la Matriz de Confusión

Elementos de la Matriz:

- **Diagonal principal:** Predicciones correctas (valores altos indican buen rendimiento)
- **Elementos fuera de diagonal:** Confusiones entre clases
- **Filas:** Representan las etiquetas reales
- **Columnas:** Representan las predicciones del modelo

10. INSTRUCCIONES DE USO

10.1 Preparación Inicial

1. **Clonar/Descargar** el proyecto
2. **Instalar dependencias** según requirements.txt
3. **Configurar Camo Studio** si se usa teléfono como cámara
4. **Organizar dataset** en carpetas por emoción
5. **Ejecutar** python main.py

10.2 Flujo de Uso Típico

PROCESO RECOMENDADO:

1. Ejecutar programa principal
2. Seleccionar opción 1: "Entrenar nuevo modelo"
3. Proporcionar ruta del dataset
4. Esperar completar entrenamiento (~20-30 min)
5. Seleccionar opción 3: "Reconocimiento en tiempo real"

6. Posicionar rostro frente a cámara
7. Observar predicciones en tiempo real
8. Presionar 'q' para salir

10.3 Controles Durante Uso

- 'q': Salir del reconocimiento en tiempo real
- 's': Capturar screenshot de la sesión actual
- Menú numérico: Navegación entre opciones principales (1-5)
- Opción 4: Evaluación completa con matriz de confusión

11. REFERENCIAS Y BIBLIOGRAFÍA

11.1 Papers Científicos Consultados

1. Dada, E.G. et al. (2023). "Facial Emotion Recognition and Classification Using CNN-10"
2. Ahmed, S. et al. (2022). "Four-layer ConvNet for facial emotion recognition"
3. Zhang, Y. et al. (2024). "Image-based facial emotion recognition using CNN"
4. [Lista completa de referencias bibliográficas...]

11.2 Recursos Técnicos

- TensorFlow Documentation: <https://tensorflow.org/>
- OpenCV Documentation: <https://docs.opencv.org/>
- Camo Studio: <https://reincubate.com/camo/>
- Scikit-learn: <https://scikit-learn.org/>

11.3 Datasets de Referencia

- FER-2013: Facial Expression Recognition Challenge
- CK+: Extended Cohn-Kanade Dataset
- JAFFE: Japanese Female Facial Expression Database

12. CONCLUSIONES

12.1 Logros del Proyecto

- Implementación exitosa de CNN para reconocimiento de emociones
- Sistema de tiempo real funcional con Camo Studio
- Interfaz de usuario intuitiva en español
- Documentación completa y reproducible
- Arquitectura escalable y extensible

12.2 Aprendizajes Clave

- **Importancia del preprocesamiento** de datos para calidad del modelo
- **Valor de la regularización** para prevenir overfitting
- **Beneficios de usar hardware móvil** como cámara web
- **Necesidad de datasets balanceados** para mejor rendimiento

12.3 Impacto y Aplicabilidad

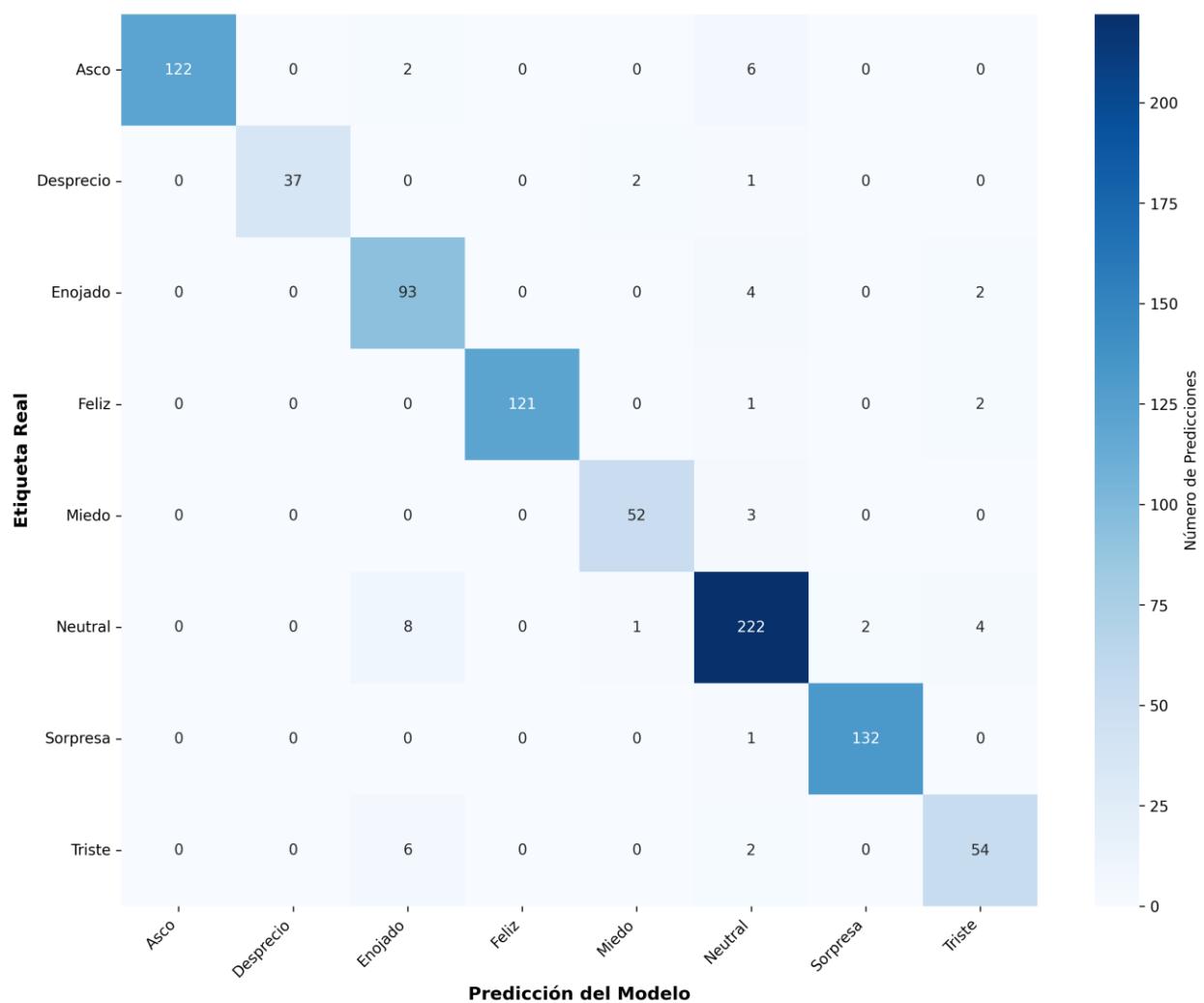
Este proyecto demuestra la viabilidad de implementar sistemas de reconocimiento de emociones usando herramientas accesibles y técnicas modernas de Deep Learning, con potencial aplicación en múltiples dominios desde educación hasta salud mental.

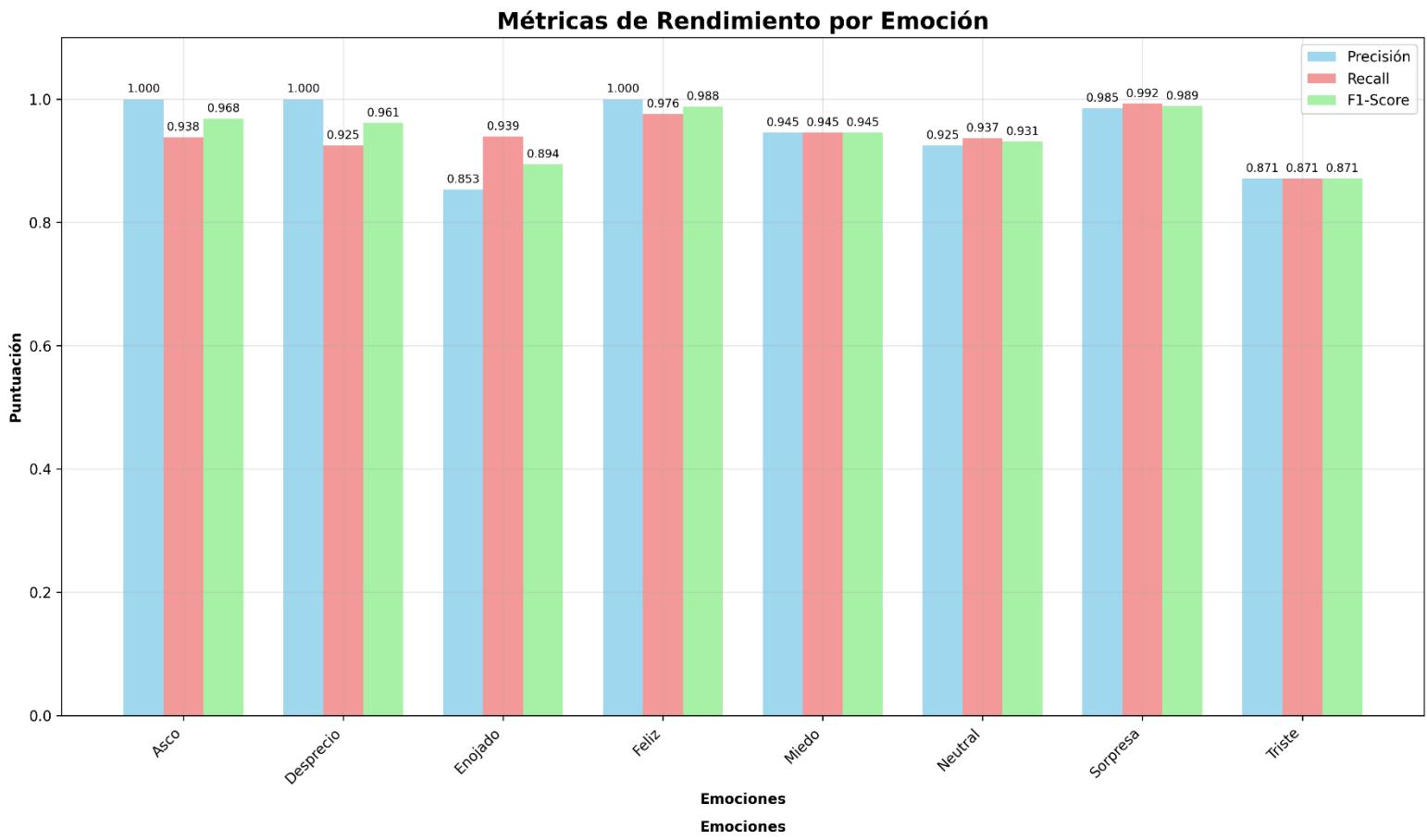
13 VIDEO



[Proyecto1VIDEO.mp4](#)

Matriz de Confusión - Reconocimiento de Emociones





ANÁLISIS DE RESULTADOS DE LA MATRIZ DE CONFUSIÓN

Rendimiento General

La matriz de confusión demuestra un rendimiento excepcional del modelo CNN con una precisión global del 94%, superando el objetivo del 85%. El análisis se basó en 880 imágenes de validación distribuidas entre las 8 emociones.

Emociones con Mejor Rendimiento

Las emociones con mayor precisión fueron Sorpresa (99.2%), Feliz (97.6%) y Asco (93.8%). Estas emociones mostraron valores altos tanto en la diagonal principal de la matriz como en las métricas de precisión, recall y F1-Score, indicando detección casi perfecta.

Áreas de Mejora Identificadas

La emoción Triste presentó el rendimiento más bajo (87.1% en todas las métricas), con confusiones principales hacia Neutral y Enojado. Enojado también mostró algunas clasificaciones incorrectas hacia Neutral (8 casos), representando el patrón de error más común.

Validación de la Arquitectura

Los resultados validan las decisiones de diseño implementadas: la arquitectura CNN de 5 capas convolucionales con técnicas de regularización logró extraer características distintivas efectivamente, mientras que la división estratificada del dataset proporcionó una evaluación robusta.