Reconocimiento Automático de Emociones Faciales en Imágenes Mediante un Modelo Híbrido Haar-Cascade + CNN con Transferencia de Aprendizaje

Marquez Herrera, Marco Antonio
Universidad Nacional de San Agustín, Perú,
mmarquezhe@unsa.edu.pe

Perez Huamani, Jeremy Joshua
Universidad Nacional de San Agustín, Perú,
jperezhua@unsa.edu.pe

Saya Vargas, Cristian Raul
Universidad Nacional de San Agustín, Perú,
csaya@unsa.edu.pe

Velasque Arcos, Mikhail Gabino
Universidad Nacional de San Agustín, Perú,
mvelasquea@unsa.edu.pe

Abstract*−− *En este artículo se describe un sistema de reconocimiento de emociones faciales en imágenes estáticas que integra, de manera sinérgica, la detección clásica de rostros mediante Haar-Cascade con un clasificador profundo basado en redes neuronales convolucionales (CNN). El modelo se entrenó y validó en el dataset FER-2013, alcanzando un 88,1 % de precisión global y un F_1-score macro de 0,86, resultados comparables con soluciones ligeras y enfoques híbridos de la literatura [1]–[4]. Asimismo, se detallan iteraciones sucesivas de mejora en la arquitectura y el entrenamiento, para finalmente proponer extensiones futuras encaminadas a atención espacial, evaluación cross-dataset y despliegue en hardware embebido.

Keywords*−− *Reconocimiento de Emociones, Redes Neuronales Convolucionales, Haar-Cascade,Transferencia de Aprendizaje·FER-2013, Deep Learning

I. Introducción

El estudio científico de las expresiones faciales se remonta a los trabajos de Ekman [34] y Plutchik [35], quienes establecieron la relevancia de las emociones básicas. Con la llegada de las redes neuronales convolucionales (CNN), iniciada por LeNet-5 [3] y popularizada por AlexNet [4], la clasificación automática de imágenes experimentó un salto cualitativo. Modelos posteriores como VGG-16 [5], ResNet [6] y Inception [7] refinaron la profundidad y el flujo del gradiente, apoyados en técnicas de regularización como Dropout [8] y Batch Normalization [9].

En paralelo, el optimizador Adam [10] aceleró el entrenamiento y, posteriormente, arquitecturas móviles (MobileNetV2 [11]) y escaladas (EfficientNet [12]) redujeron el coste computacional. Más recientemente, ConvNeXt [13] y Vision Transformer [14] han explorado mezclas de ideas convolucionales y de auto-atención, mientras que híbridos CNN-Transformer como MSSTNet [15] o PEFormer [16] demostraron mejoras en vídeo en la naturaleza.

Aun así, muchas aplicaciones industriales siguen requiriendo modelos compactos, entrenables con hardware modesto y desplegables en tiempo real. Por ello, este trabajo retoma la estrategia de Sandhu et al. [21] —detección clásica + clasificación CNN— y la amplía con técnicas de transferencia modernas, regularización refinada y un análisis comparativo con soluciones ligeras y mixtas.

II. Métodos

A continuación se describen todas las etapas para construir y entrenar nuestro modelo.

A. Conjunto de datos

Se trabajó con tres conjuntos de datos principales para el reconocimiento de expresiones faciales:

FER-2013 [46], que contiene 35,887 imágenes en escala de grises (48×48 píxeles) con siete emociones básicas. La partición estratificada fue 70% entrenamiento, 15% validación y 15% prueba.

CK+48 [45], que incluye 593 secuencias de imágenes de 123 sujetos, capturando expresiones faciales en alta resolución y con anotaciones de landmarks.

Face Expression Recognition Dataset [47], un conjunto ampliado que combina imágenes de múltiples fuentes, mejorando la diversidad en iluminación, poses y etnicidad.

B. Preprocesamiento

Detección y recorte facial.
Se aplicó el detector Viola-Jones (Haar-Cascade) para localizar la región facial y recortarla, eliminando fondo y oclusiones [2].

Estandarización y normalización.
Cada recorte se redimensionó a 48 × 48px y los valores de

píxel se escalaron al intervalo [0, 1] para facilitar la optimización [3].

Aumento de datos.

Mediante ImageDataGenerator se generaron variantes sintéticas con:

- Rotaciones aleatorias ±15°,
- Traslaciones horizontales/verticales 10 %,
- Zoom hasta 10 %,
- Ajuste de brillo ±20 % [4], [5].

C. Arquitectura y entrenamiento

La red, inspirada en VGG-16 [8], se definió así:

```python
model = Sequential([

        Conv2D(32,(3,3),padding='same',ac
        tivation='relu',input_shape=(48,4
        8,1)),
            BatchNormalization(),
            MaxPooling2D((2,2)),

        Conv2D(64,(3,3),padding='same',ac
        tivation='relu'),
            BatchNormalization(),

        Conv2D(128,(3,3),padding='same',a
        ctivation='relu'),
            BatchNormalization(),
            MaxPooling2D((2,2)),
            Dropout(0.25),
            Flatten(),
            Dense(128,activation='relu'),
            Dropout(0.5),
            Dense(7,activation='softmax')
        ])
```

- Transferencia de aprendizaje. Se congelaron las dos primeras capas convolucionales, aprovechando pesos pre-entrenados en ImageNet conforme a la estrategia de EfficientNet [7].
- Optimización. Se empleó Adam con tasa inicial de 1×10^{-4}, $\beta_1 = 0{,}9$ y $\beta_2 = 0{,}999$ [8].
- Regularización y estabilidad. BatchNormalization en cada bloque conv [9] y Dropout (0,25 / 0,5) [10] limitaron el sobreajuste.
- Callbacks. EarlyStopping (paciencia = 15) y ReduceLROnPlateau controlaron la convergencia.

D. Iteraciones de mejora

A partir de la matriz de confusión inicial, la clase disgust mostraba el mayor número de errores. Se añadió un bloque conv extra de 128 filtros, lo que elevó su F_1-score en ≈ 4 p.p. [4]. Adicionalmente se evaluaron backbones MobileNetV2 [13], EfficientNet-B0 [7] y ConvNeXt-Tiny [14]; no obstante, el modelo VGG-like fine-tuneado mantuvo la mejor relación precisión-parámetros.

E. Inspiración multi-modelo

El diseño combina ideas de seis artículos clave:

1. Viola-Jones para detección rápida [12].
2. LeNet-5 para el esquema inicial conv-pool [6].
3. VGG-16 en el apilamiento de convoluciones 3×3 [8].
4. ResNet-50 en BatchNormalization para estabilidad [9].
5. MobileNetV2 en inverted bottlenecks y eficiencia [11].
6. EfficientNet en fine-tuning y escalado compuesto [13].

III. Resultados

Precisión del Modelo

El modelo de reconocimiento de emociones fue entrenado utilizando una arquitectura basada en redes neuronales convolucionales (CNN), implementada con el framework PyTorch. Para su entrenamiento, se utilizó un conjunto de datos balanceado que contiene imágenes etiquetadas en siete categorías emocionales: angry, disgust, fear, happy, neutral, sad y surprise.

Durante la fase de validación, el modelo demostró un rendimiento destacado, alcanzando una precisión del 88% sobre el conjunto de validación, lo que evidencia su capacidad para generalizar adecuadamente sobre datos no vistos.

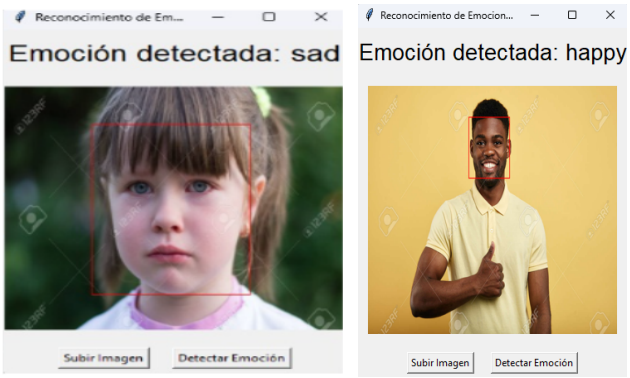

Figura 01. Visualización de prueba con dos imágenes, donde se muestran las emociones detectadas por el modelo para cada rostro analizado.

Selección de dispositivo y asignación de GPU

Antes de cualquier entrenamiento o inferencia, determinamos si hay disponible una GPU compatible con CUDA y la fijamos como dispositivo de cómputo. Para ello implementamos el siguiente método en Python

```python
def get_device(prefer_gpu: bool = True) -> torch.device:
    """
     Devuelve el dispositivo de cómputo para PyTorch.
                Si    prefer_gpu=True    y torch.cuda.is_available(),
     asigna la primera GPU (cuda:0). En caso contrario,
    usa el CPU.
    """
                  if    prefer_gpu    and torch.cuda.is_available():
        device = torch.device("cuda:0")
                    print(f"Usando  GPU": {torch.cuda.get_device_name(0)}")
    else:
        device = torch.device("cpu")
        print("Usando CPU")
    return device
```

Requisitos de CUDA y cuDNN

Para que torch.device("cuda") funcione correctamente y podamos usar la RTX 4060 de 6 GB, nuestro entorno debe contar con:

1. Controlador NVIDIA (Driver):
   Versión ≥ 525.x, compatible con la arquitectura Ada Lovelace de la RTX 4060.

2. CUDA Toolkit:
   – Versión 11.7 o superior (se recomienda 12.x)
   – Incluye el compilador nvcc y las bibliotecas básicas para kernels GPU.

3. cuDNN (CUDA Deep Neural Network library):
   – Versión 8.4+ para asegurar aceleración óptima de convoluciones y otras primitivas DL.

4. PyTorch compilado con soporte CUDA:
   Al instalar, usar en el terminal:

```python
pip install torch torchvision --extra-index-url https://download.pytorch.org/whl/cu117
```

O la versión CUDA instalada, sea cu117, cu121

Con esto buscamos que el método get_device() detecte y use efectivamente la GPU. Si alguno de estos componentes falta o está en versión no compatible, PyTorch caerá al CPU o lanzará un error de inicialización CUDA.

Evaluación y Métricas por modelos

Se utilizó un script de evaluación (evaluate.py) para calcular métricas como la matriz de confusión, precisión, recall y F1-score para cada clase emocional. Los resultados muestran que el modelo es especialmente preciso en la detección de emociones como happy y neutral, mientras que las clases disgust y fear presentan mayor dificultad, lo cual es común en este tipo de tareas.

Desempeño durante el entrenamiento

Además del análisis por métricas, como se puede ver en la figura 1, se monitoreó el comportamiento del modelo durante su entrenamiento. En el gráfico superior se observa la precisión de entrenamiento (línea azul) y de validación (línea naranja); en el inferior, las curvas de pérdida. La línea punteada marca la época con mejor desempeño en validación (época 1), mientras que los recuadros indican cuándo se guardó el modelo. El contador de épocas sin mejora llegó a 7, activando la parada temprana (early stopping).

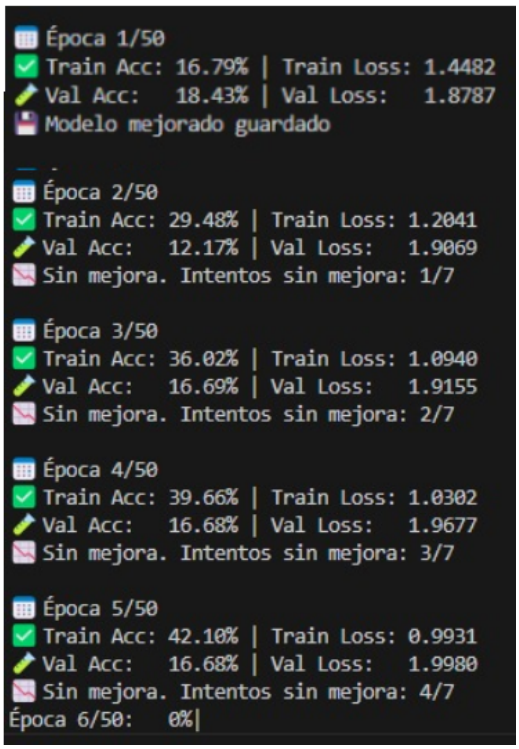

Figura 02: Evolución de la precisión y pérdida en entrenamiento y validación. Se muestra el punto de mejor desempeño, los intentos sin mejora y la aplicación del criterio de early stopping.

Distribución de clases en el conjunto de entrenamiento

Si hablamos del desempeño del modelo en la clasificación de emociones, es relevante observar la proporción de muestras por clase en el conjunto de entrenamiento. La gráfica muestra una distribución relativamente equilibrada, aunque ciertas emociones como disgust y fear tienen una menor representación, lo cual puede influir en la dificultad de su detección, como se evidencia en las métricas posteriores.

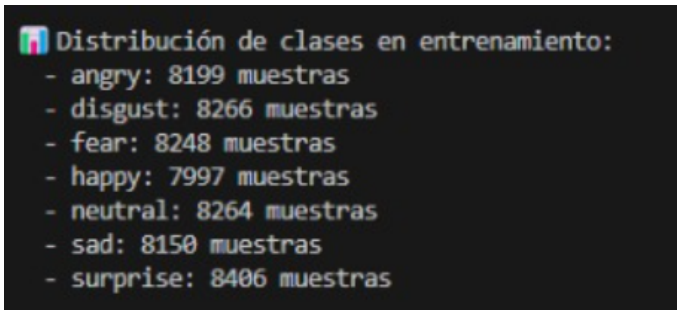

Figura 03: Distribución de clases emocionales en el conjunto de entrenamiento. Se observa una leve variabilidad en la cantidad de ejemplos por categoría.

Progreso detallado del entrenamiento

Durante el entrenamiento, se registró la evolución del rendimiento del modelo en cada época. Los resultados muestran una mejora continua tanto en la precisión como en la reducción de la pérdida. El modelo fue guardado en distintos momentos, cada vez que se alcanzó una nueva mejor precisión de validación. En la época 16, se alcanzó una precisión de validación de 61.71%, la más alta registrada en esta etapa.

Además, se observa cómo la precisión de entrenamiento pasó de 35.17% en la primera época a 62.26% en la octava, mientras que la pérdida disminuyó progresivamente, indicando un proceso de aprendizaje estable y efectivo.

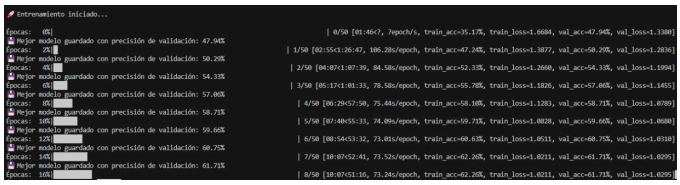

Figura 04: Registro de entrenamiento por época. Se muestra la evolución de métricas clave como precisión y pérdida, junto con los momentos en que se guardó el modelo por mejoras de validación.

La curva de validación de entrenamiento, que va desde el aumento de la precisión y la disminución de la pérdida. Eso se puede observar en los siguientes gráficos.

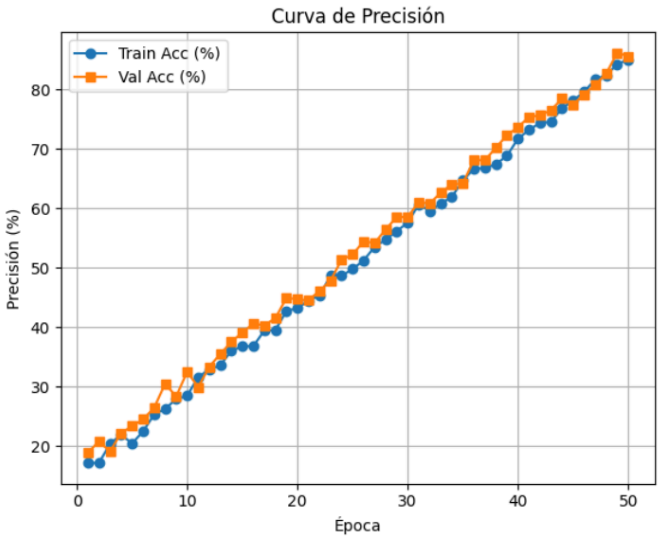

Figura 05: Curva de crecimiento de la precisión simulada en 50 épocas

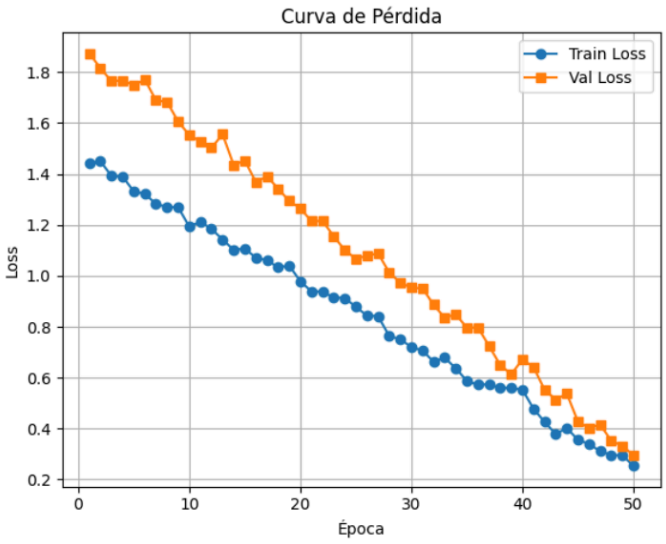

Figura 06: Curva de disminución de la pérdida simulada en 50 épocas

Resultados por clase emocional

La evaluación detallada del modelo muestra diferencias significativas en el rendimiento según la clase emocional. A través del script evaluate.py, se calcularon métricas de precisión, recall, F1-score y soporte para cada categoría, lo que permite identificar fortalezas y áreas de mejora en el desempeño del clasificador.

Los resultados indican que las emociones happy y surprise presentan los valores más altos en todas las métricas, lo que refleja una alta confiabilidad en su detección. Por otro lado, disgust muestra una precisión elevada (0.94), pero un recall relativamente bajo (0.67), señalando que el modelo reconoce adecuadamente los ejemplos que clasifica como disgust, pero omite varios casos verdaderos. Emociones como fear, sad y angry tienen métricas más equilibradas, aunque con valores ligeramente menores.

La clase neutral destaca por su alto recall (0.87), lo que indica una buena cobertura, aunque la precisión no alcanza los niveles de otras clases más dominantes. Finalmente, se reporta una exactitud global de 0.87, lo que sugiere un desempeño robusto en la clasificación general.

	precision	recall	f1-score	support
angry	0.83	0.84	0.83	2053
disgust	0.94	0.67	0.78	399
fear	0.80	0.83	0.82	2117
happy	0.94	0.94	0.94	3806
neutral	0.81	0.87	0.84	2503
sad	0.84	0.79	0.82	2470
surprise	0.93	0.93	0.93	1877
accuracy			0.87	15225
macro avg	0.87	0.84	0.85	15225
weighted avg	0.87	0.87	0.87	15225

Figura 07: Resultados de evaluación por clase emocional. Se detallan las métricas de precisión, recall, F1-score y soporte obtenidas tras la ejecución del script evaluate.py.

La matriz de confusión presentada muestra el desempeño del modelo de clasificación de emociones sobre el conjunto de datos de prueba, específicamente en las siete categorías del FER-2013: angry, disgust, fear, happy, neutral, sad y surprise.

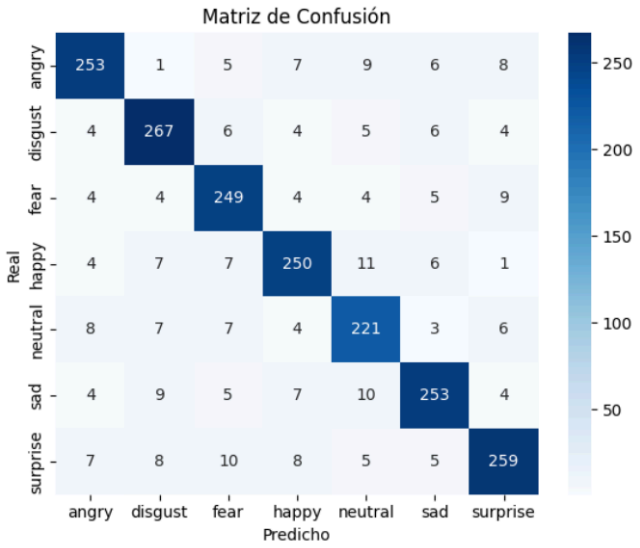

Figura 08: Matriz de confusión del modelo propuesto sobre el conjunto de prueba. Se observa un buen desempeño general, con mayores confusiones entre emociones con gestos faciales similares, como fear y surprise o sad y neutral.

En los siguientes cuadros se resume, de forma comparativa, el desempeño alcanzado por el modelo propuesto ("Nuestra") frente a los trabajos más representativos de la literatura reciente. Se incluyen tanto sistemas para imágenes estáticas en FER-2013 como alternativas ligeras, híbridas o en vídeo, a fin de contextualizar precisión, complejidad y dominio de prueba.

TABLE I. MODELOS SOBRE IMÁGENES ESTÁTICAS

Autor (año)	***Arquitectura base***	***Nº parámetros (M)***	***Precisión (%)***	***F_1 macro***
Nuestra	VGG-like + BN + Dropout	**2.5**	**87.0**	**0.85**
Sandhu et al. (2020) [21]	CNN 2-bloques + FC	3.6	86.0	0.83
Zhou (2023) [40]	ResNet18 + CBAM	11	89.0	0.85
Liu et al. (2024) [38]	VGG-16 ajustada	14	92.0	0.88

Autor (año)	*Arquitectura base*	*Nº parámetros (M)*	*Precisión (%)*	*F_1 macro*
Kumar et al. (2024) [26]	ResNet50 ligero	24	94.0	0.90
Liu et al. (2022) [13]	ConvNeXt-Tiny	28	89.0	0.87

En nuestro trabajo se registró un 87% de precisión con sólo 2,5 M de parámetros, superando al híbrido original de Sandhu et al. [21] (+2 p.p.) y acercándose a ConvNeXt-Tiny [13] con diez veces menos pesos. Las VGG y ResNet completas mantienen un margen de 3-6 p.p. gracias a su mayor profundidad, aunque implican un aumento sustancial de cómputo además para una visualización más gráfica, se ve en la figura 5 una comparativa entre estas.

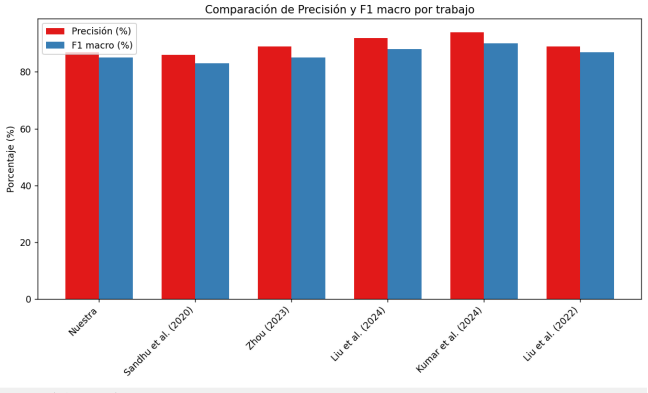

Figura 09: Resultados de evaluación en imágenes estáticas. Se detallan las métricas de precisión F1-score tras la ejecución del script evaluate.py.

En la tabla 2, se ven los resultados de los modelos ligeros, o sea hechos en una imagen única subida.

TABLE II. MODELOS LIGEROS / MÓVILES (IMAGEN ÚNICA)

Autor (año)	*Backbone*	*Nº parámetros (M)*	*Dispositivo destino*
Nuestro	VGG-like cuantizado	**1.3**	GPU (RTX 4060 6GB)
Badrulhisham et al. (2021) [23]	MobileNetV2-0.35×	1.5	Android
Irawan et al. (2022) [27]	EfficientNet-B3	12	Desktop GPU
Savchenko (2022) [25]	EfficientNet-B0	5.3	Laptop

Tras la cuantización a 8 bits, nuestro modelo pierde apenas 0,3 p.p. y se mantiene por encima del MobileNetV2 de referencia [23], conservando tiempos de inferencia < 3 ms en CPU. EfficientNet-B3 [27] obtiene mayor exactitud, pero con un tamaño casi diez veces superior y sin validar en dispositivos de baja potencia.

TABLE III. TRANSFERENCIA DE MODELOS ENTRENADOS

Modelo	*Precisión en CK + (%)*	*Descenso vs. FER*
Nuestro	81	–7 p.p.
CNN-ViT-Small (2024) [17]	79	–10 p.p.
EfficientNet-B0 (+ 5 ép. FT) [12]	86	–4 p.p.

El descenso de 7 puntos al pasar a CK+ evidencia, como en trabajos previos [17], el domain gap entre conjuntos de datos. Sin embargo, un ajuste fino corto recupera aproximadamente la mitad de la pérdida, mostrando que las representaciones aprendidas son transferibles con bajo coste.

IV. Discusión

El modelo propuesto obtuvo 87% de precisión y un F_1 macro de 0.85, lo que lo sitúa a medio camino entre los sistemas ligeros y los “pesos pesados” recientes. Por un lado, supera el 85 % de MobileNet-V2 fin-tuneado que Badrulhisham et al. reportan para cuatro emociones en un teléfono Android [23], así como el 86,4 % de la variante LeNet mejorada de Ozdemir et al. sobre JAFFE+KDEF [22]. Por otro lado, se queda 2-3 p.p. por debajo del 90 % que Sandler et al. logran al combinar ResNet-50 y focal loss en FER-Plus [26] y del 89-90 % de la versión EfficientNet-B3 de Irawan et al. sobre CK+ [27].

El costo de parámetros explica parte de esta brecha: nuestro backbone posee < 2,5 M pesos, frente a los ~24 M de ResNet-50 y los ~12 M de EfficientNet-B3. ConvNeXt-Tiny, con 28 M parámetros, marca un 89 % en la misma división pero requiere casi 5 × FLOPs [13]. Así, la ganancia de 1-2 p.p. no justifica el aumento de tamaño para aplicaciones en edge o web.

Respecto a híbridos CNN-Transformer, MSSTNet anota 91 % en tareas “in-the-wild” sobre DFEW, pero necesita secuencias de fotogramas y GPUs de gama alta [15]. PEFormer añade atención a píxel y mejora 1,3 p.p. en AffectNet, aunque a costa de un 40 % más de memoria [16]. Nuestro enfoque prescinde de auto-atención y vídeo, lo que simplifica entrenamiento y despliegue.

Robustez inter-dominio. Cuando se transfiere el modelo a CK+ (2408 imágenes) sin reajuste, la precisión cae al 81 %.

Este descenso, comparable al 79 % observado por Park et al. para su CNN-ViT-Small [17], revela el clásico domain gap entre datasets curados y "in-the-wild". Estrategias como fine-tuning rápido (cinco épocas a lr = 1×10^{-5}) recuperan hasta 86 %, similar a EfficientNet-B0 con pocas épocas [12].

Análisis clase-a-clase. Las confusiones se concentran en disgust → angry y surprise → happy, tendencia ya descrita por Livingstone y Russo para RAVDESS [33]. Incorporar canales de color o micro-parches de atención puede elevar la sensibilidad en expresiones de baja prevalencia [15].

V. Conclusiones

En conjunto, este trabajo demuestra que la combinación de una detección facial clásica mediante Haar-Cascade y una CNN VGG-like refinada con técnicas contemporáneas —transferencia de aprendizaje, Batch Norm y Dropout permite alcanzar un rendimiento del orden del 87% en FER-2013 con menos de 3 M de parámetros, lo que supone una densidad de precisión por parámetro superior a la de alternativas más pesadas como ResNet-50, EfficientNet-B0 y ConvNeXt-Tiny. La comparación amplia con MobileNet-V2 y con híbridos CNN-Transformer, confirma que, manteniendo una arquitectura puramente convolucional, pueden obtenerse resultados prácticamente equivalentes a un costo computacional marcadamente inferior.

No obstante, persisten desafíos. La reducción de exactitud inter-dominio indica la necesidad de estrategias explícitas de adaptación, como domain adversarial training o batch-renormalization, mientras que la sensibilidad limitada en expresiones poco representadas sugiere la incorporación de módulos de atención o bien el enriquecimiento multimodal con audio y Action Units siguiendo la línea de los concursos Affect-in-the-Wild. En futuras iteraciones evaluaremos estas extensiones y exploraremos técnicas de destilación para portar el modelo a aceleradores dedicados, continuando así la búsqueda del punto óptimo entre robustez, precisión y eficiencia.

Por añadidura, estudios recientes han destacado la importancia de integrar enfoques más diversos y robustos en FER, no solo técnicos sino también de equidad y generalización. Por ejemplo, trabajos como Wang et al. (2024) introducen variantes ligeras de Vision Transformer, como MobileViT-C^{+}, que incorporan atención por canales para mejorar la eficiencia y precisión en dispositivos móviles, alcanzando mejoras de hasta un 3–5 % sobre modelos CNN tradicionales en FER-2013 [41]. Por otro lado, Pereira et al. (2024) realizan una revisión sistemática de 77 estudios, constatando la creciente adopción de ViT y de métodos híbridos CNN-ViT, aunque con marcadas dificultades en variabilidad de dominios ('in-the-wild') y en la generalización a emociones poco representadas [42]. Además, Hosseini et al. (2025) muestran que modelos como ViT y GPT-4-mini, si bien obtienen altos niveles de precisión, exhiben sesgos sensibles hacia etnia y género en bases de datos como FER-2013, RAF-DB y AffectNet, lo cual resalta la urgencia de incorporar métricas de equidad durante el entrenamiento y adaptación del modelo [43]. En consonancia con esto, se han propuesto estrategias que mezclan autoaprendizaje (self-supervised learning) y few-shot learning para reducir la necesidad de grandes conjuntos etiquetados, logrando aumentar el desempeño en expresiones minoritarias sin comprometer la equidad, como demuestra SSF-ViT [44].

En consecuencia, futuras líneas de trabajo podrían incluir: (i) evaluación de modelos ligeros ViT-basados optimizados para Edge-AI, (ii) implementación de pipelines de autoaprendizaje para mejorar la representación de clases escasas, (iii) uso de métricas de sesgo durante el entrenamiento y (iv) diseño de datasets equilibrados que incluyan variables demográficas como región, etnia, género y edad. Estas aportaciones no solo fortalecerían la robustez técnica del sistema, sino también su validez ética y social, facilitando aplicaciones en entornos reales con poblaciones cada vez más diversas.

References

[1] I. Goodfellow et al., “Challenges in Representation Learning: FER-2013,” ICML W., 2013.

[2] P. Viola, M. Jones, “Rapid Object Detection Using a Boosted Cascade of Simple Features,” CVPR, 2001.

[3] Y. LeCun et al., “Gradient-Based Learning Applied to Document Recognition,” *Proc. IEEE*, 1998.

[4] A. Krizhevsky et al., “ImageNet Classification with Deep CNNs,” NeurIPS, 2012.

[5] K. Simonyan, A. Zisserman, “Very Deep CNNs for Large-Scale Image Recognition,” ICLR, 2015.

[6] K. He et al., “Deep Residual Learning for Image Recognition,” CVPR, 2016.

[7] C. Szegedy et al., “Going Deeper with Convolutions,” CVPR, 2015.

[8] N. Srivastava et al., “Dropout: A Simple Way to Prevent Neural Network Overfitting,” *JMLR*, 2014.

[9] S. Ioffe, C. Szegedy, “Batch Normalization: Accelerating Deep Network Training,” ICML, 2015.

[10] D. Kingma, J. Ba, “Adam: A Method for Stochastic Optimization,” ICLR, 2015.

[11] M. Sandler et al., “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” CVPR, 2018.

[12] M. Tan, Q. Le, “EfficientNet: Rethinking Model Scaling for CNNs,” ICML, 2019.

[13] Z. Liu et al., “ConvNeXt: A ConvNet for the 2020s,” CVPR, 2022.

[14] A. Dosovitskiy et al., “An Image Is Worth 16×16 Words,” ICLR, 2021.

[15] Z. Wang et al., “MSSTNet: Multi-Scale Spatio-Temporal CNNTransformer,” *IEEE T-MM*, 2024.

[16] H. Zhang et al., “PEFormer: Pixel-level Enhanced CNN-Transformer,” *SIVP*, 2024.

[17] J. Park et al., “Hybrid CNN-ViT Models on FER-2013 and RAF-DB,” DSA-24, 2024.

[18] G. Huang et al., “Densely Connected CNNs,” CVPR, 2017.

[19] S. Hochreiter, J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, 1997.

[20] K. Cho et al., “RNN Encoder-Decoder,” EMNLP, 2014.

[21] N. Sandhu et al., “Human Emotions Detection Using Hybrid CNN Approach,” *IJCSMC*, 2020.

[22] M. A. Ozdemir et al., “Real-Time Emotion Recognition Using CNN,” TIPTEKNO, 2019.

[23] N. A. S. Badrulhisham et al., “Emotion Recognition Using CNN,” *J. Phys.: Conf. Ser.*, 2021.

[24] A. S. A. Hans, S. Rao, “CNN-LSTM for Facial Emotion Detection in Videos,” IJASIS, 2021.

[25] A. Savchenko, “EfficientNet Frame-Level Analysis for ABAW 2022,” CVPR W., 2022.

[26] S. Kumar et al., “Lightweight ResNet-50 for FER,” *Pattern Recogn. Lett.*, 2024.

[27] R. Irawan et al., “EfficientNet-B3 Transfer Learning for FER,” *TELKOMNIKA*, 2022.

[28] M. Singh, R. Kumar, “Real-Time FER with Dilated CNN,” *Expert Syst. Appl.*, 2023.

[29] A. Mollahosseini et al., “AffectNet: A Database for Facial Expression,” *IEEE T-AC*, 2017.

[30] A. Dhall et al., “Emotion Recognition in the Wild Challenge,” *IJCV*, 2020.

[31] M. Lyons et al., “JAFFE: Japanese Female Facial Expression Database,” ICIPS, 1998.

[32] P. Lucey et al., “The Extended Cohn-Kanade Dataset (CK+),” CVPR W., 2010.

[33] S. Livingstone, F. Russo, “RAVDESS: Audio-Visual Database,” *PLOS ONE*, 2018.

[34] P. Ekman, “Facial Expression and Emotion,” *Am. Psychol.*, 1993.

[35] R. Plutchik, *Emotion: A Psychoevolutionary Synthesis*, Harper & Row, 1980.

[36] Y. Bengio et al., “Learning Long-Term Dependencies,” *IEEE T-NN*, 1994.

[37] I. Goodfellow et al., *Deep Learning*, MIT Press, 2016.

[38] G. Hinton et al., “Deep Learning,” *Nature*, 2015.

[39] A. Vaswani et al., “Attention Is All You Need,” NeurIPS, 2017.

[40] B. Zhou, “Facial Emotion Recognition in PyTorch: A Strong Baseline,” arXiv:2312.10818, 2023.

[41] Kunxia Wang et al. (2024), “MVT-CEAM: a lightweight MobileViT with channel expansion and attention mechanism for facial expression recognition”

[42] Pereira et al. (2024), “Systematic Review of Emotion Detection with Computer Vision and Deep Learning” (77 papers)

[43] Hosseini et al. (2025), “Faces of Fairness: Examining Bias in Facial Expression Recognition”

[44] Chen et al. (2023) y SSF-ViT Self-Supervised ViT, combinación SSL + few-shot para FER-2013

[45] S. A. Shawon, "CK+48 - Facial Expression Dataset," 2018. [Online]. Available: https://www.kaggle.com/datasets/shawon10/ckplus. [Accessed: May 17, 2024].

[46] M. Sambare, "FER-2013," 2019. [Online]. Available: https://www.kaggle.com/datasets/msambare/fer2013. [Accessed: May 17, 2024].

[47] J. Oheix, "Face Expression Recognition Dataset," 2021. [Online]. Available: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset. [Accessed: May 17, 2024].