# MicroLightCNN

## Fundamentos, Arquitectura y Despliegue en ESP32

Dr. Hugo Arnoldo Mitre Hernández

CIMAT Zacatecas · Julio 2026

*Proyecto: ¡Jarvis, Estoy Molesto!*

# Motivación y Contexto

## Problema

La violencia doméstica afecta a millones de familias en México y el mundo. El enojo verbal frecuentemente precede a la agresión física.

Se necesita un sistema de monitoreo:

- Compacto y de bajo costo
- Funcional en tiempo real
- Ejecutable en microcontrolador (sin nube)
- Capaz de identificar quién habla

### Restricciones del ESP32-S3

- 8 MB PSRAM total
- Sin GPU, dual-core a 240 MHz
- TFLite Micro: operadores limitados
- Presupuesto de parámetros < 50k
- Pipeline dual (emoción + speaker)

# ¿Qué es una Red Neuronal Convolucional (CNN)?

*Las CNN aprenden características espaciales jerárquicas automáticamente.*

1

## Convolución

Aplica filtros (kernels) que recorren la entrada detectando patrones locales como bordes, texturas y formas.

2

## Pooling

Reduce dimensiones espaciales preservando las características más importantes. MaxPool selecciona el valor máximo de cada región.

3

## Clasificación

Capas densas (fully connected) combinan las características extraídas para producir la predicción final (ej: Angry vs. Not-Angry).

# Operación de Convolución

$$S(i, j) = \Sigma_m \Sigma_n I(i+m, j+n) \cdot K(m, n)$$

**I(i+m, j+n)** valor del píxel en la posición (i+m, j+n) de la imagen de entrada

**K(m, n)** kernel o filtro convolucional (ej: 3×3 valores aprendidos)

**S(i, j)** valor de salida en el mapa de características (feature map)

**Σ** sumatoria sobre la región local cubierta por el kernel

## Interpretación

El kernel actúa como un detector de patrones. Se multiplica elemento a elemento con la región de la imagen y se suman los resultados.

**Valores altos → patrón detectado**

**Valores bajos → región uniforme**

# LightCNN Original y Max Feature Map (MFM)

## LightCNN (Wu et al., 2018)

Diseñada originalmente para reconocimiento facial a gran escala con etiquetas ruidosas.

### Innovación clave: MFM (Max Feature Map)

Dada una entrada $X \in \mathbb{R}^{HxWx2c}$, MFM divide los canales en dos grupos de C y retiene el máximo elemento a elemento:

$$MFM(X)_{i,j,k} = \max( X_{i,j,k} , X_{i,j,k+}{}^{c} ) \quad k = 1,\ldots,C$$

- Selecciona el canal más activo entre dos filtros
- Promueve representaciones sparse
- Reduce canales a la mitad (96→48)

### Ejemplo Numérico de MFM

**Entrada $x_1$**

```
[ 1   5 ]
[ 3   2 ]
```

```
[ 4   2 ]
[ 1   6 ]
```

**Entrada $x_2$**

**Salida MFM = max($x_1$, $x_2$):**

```
[ 4   5 ]
[ 3   6 ]
```

*Se conserva solo el valor más alto de cada par. Actúa como selección automática de la característica más relevante.*

# De LightCNN a MicroLightCNN: Adaptación para MCU

## Incompatibilidades de LightCNN-9

**1. MFM requiere Split + Maximum**

Operadores ausentes en el micro-interpreter resolver de TFLite Micro.

**2. Nueve capas convolucionales**

Sobre espectrogramas de 20×63, las dimensiones espaciales se saturan después de 3 etapas de pooling.

**3. PSRAM insuficiente**

El footprint resultante excluiría el modelo de speaker simultáneo.

## Principios de MicroLightCNN

**Se preservan:**

- Uso exclusivo de kernels 3×3
- Max-pooling agresivo no-overlapping antes de la cabeza FC

**Se reemplazan:**

- MFM → ReLU (soportado nativamente en TFLite-M)
- 9 capas → 3 bloques convolucionales
- Se agrega BatchNorm para estabilizar cuantización

# Extracción de Características: MFCC


## Parámetros de configuración

**Frecuencia de muestreo:** 16 kHz
**Ventana (WS):** 512 muestras (32 ms)
**Hop (OS):** 256 muestras (50% overlap)
**Bandas Mel:** 20 filtros triangulares
**Rango frecuencia:** 20 Hz – 8 kHz

### Salida: $M \sim \mathbb{R}^{20x63}$

20 coeficientes cepstrales (bandas de frecuencia Mel) × 63 frames temporales.

Esta misma matriz MFCC alimenta ambas ramas del pipeline dual: MicroLightCNN (emoción) y MLP-XiEmbedding (speaker).

*Implementación idéntica en Python y C++ para consistencia.*

# Arquitectura de MicroLightCNN — Capa por Capa

| Etapa | Operación | Salida | Params |
|---|---|---|---|
| Input | MFCC tensor | 20 × 63 × 1 | 0 |
| Stem | Conv2D 3×3, 16f, BN, ReLU | 20 × 63 × 16 | 160 |
| Block 1 | MaxPool 2×2 stride 2 | 10 × 31 × 16 | 0 |
| Block 2 | Conv2D 3×3, 16f, BN, ReLU + Pool | 5 × 15 × 16 | 2,320 |
| Block 3 | Conv2D 3×3, 16f, BN, ReLU + Pool | 2 × 7 × 16 | 2,320 |
| Flatten | Flatten + L2-norm | 224-d | 0 |
| FC | Dense(64) + Dropout(0.5) | 64 | 14,400 |
| Output | Dense(1) + Sigmoid | 1 | 65 |

**Totales**

**19,409**
parámetros

**59.7 KB**
arena INT8

**28.8 KB**
flash INT8

**3×3**
solo kernels

# Cuantización INT8: Menos Memoria, Más Precisión

## ¿Qué es la cuantización INT8?

Convierte los pesos y activaciones del modelo de punto flotante (Float32, 32 bits) a enteros de 8 bits (INT8).

**Reducción de memoria:** ~4× (cada valor pasa de 4 bytes a 1 byte)
**Reducción de arena:** 3.5× (210.9 KB → 59.7 KB)
**Reducción de flash:** 2.8× (81.0 KB → 28.8 KB)
*Se usa per-tensor affine quantization con un set de calibración de 100 utterances.*

### Float32 (referencia)

Accuracy: 82.8% · Arena: 210.9 KB · Flash: 81.0 KB

F1: 82.3 · AUC: 90.6

### INT8 (elegido)

**Accuracy: 83.2% (+0.4) · Arena: 59.7 KB · Flash: 28.8 KB**

F1: 82.8 · AUC: 90.5

*La cuantización MEJORA accuracy gracias al efecto regularizador del redondeo de pesos + BatchNorm folding.*

# Resultados de Evaluación — ESD Test Set

| 83.2% | 0.828 | 0.905 | 0.848 | 0.808 |
|---|---|---|---|---|
| Accuracy | F1-Score | AUC | Precision | Recall |

## Matriz de Confusión (N=1,200, balanceado)

| | Pred: Not-Angry | Pred: Angry |
|---|---|---|
| **Real: Not-Angry** | **513 (85.5%)** | 87 (14.5%) |
| **Real: Angry** | 115 (19.2%) | **485 (80.8%)** |

### Interpretación

El modelo es marginalmente conservador en τ=0.5: specificity (85.5%) > recall (80.8%).

El AUC de 0.905 indica que un ajuste fino del umbral puede recuperar recall sin re-entrenamiento.

*Dataset: ESD (Emotional Speech Database), 1,200 utterances balanceadas en test set.*

# Profiling End-to-End en ESP32-S3

| Etapa | Tiempo (ms) | % Total | Fuente |
|---|---|---|---|
| Captura I2S (2s window) | 2,000 | — | no-aditivo |
| Extracción MFCC | 190.1 | 40.8% | medido |
| MicroLightCNN inference | 219.0 | 47.0% | medido |
| Xi-Vector pooling + Scaler | 1.0 | 0.2% | estimado |
| MLP-XiEmbedding inference | 55.0 | 11.8% | estimado |
| Fusión + voto temporal | 0.5 | 0.1% | estimado |
| **TOTAL (procesamiento)** | **465.5** | **100%** | — |

**Memoria**

**Arena total:**

**81.3 KB**

MicroLightCNN: 59.7 KB

MLP-XiEmb: 21.6 KB

SRAM libre: ≥229 KB

(stack, DMA, WiFi/BLE)

**Procesamiento total < 500 ms + captura 2s = ciclo completo < 2.5 s (near real-time)**

# Evaluación en Entorno Real — Pruebas de Distancia

Se realizó un protocolo de prueba espacial radial en un entorno doméstico real para evaluar el rendimiento del sistema a diferentes distancias del micrófono.

| 50 cm | 100 cm | 150 cm | 200 cm |
|---|---|---|---|
| Distancia cercana, conversación íntima | Distancia conversacional normal | Distancia intermedia en una habitación | Distancia máxima recomendada |
| Óptimo | Bueno | Aceptable | Límite |

**Accuracy conjunta (anger + speaker): 82.8% en rango 50–200 cm**

# Hardware del Prototipo Embebido

MCU

### ESP32-S3-N16R8

MCU dual-core Xtensa LX7
@ 240 MHz
16 MB Flash, 8 MB PSRAM
WiFi + BLE integrado

MIC

### Microfono

Micrófono I2S digital
Omnidireccional
16 kHz sampling rate

LED

### Matriz LED

Display serial FC16_HW
Muestra intensidad 0-100%
Feedback visual en tiempo real

**Core 0: Inferencia (MicroLightCNN + MLP-XiEmb) | Core 1: I2S DMA, MFCC, Periféricos**

# Protocolo de Entrenamiento

## Configuración

**Framework:** TensorFlow 2.20 / Keras
**Optimizer:** Adam (lr = $10^{-3}$)
**Batch size:** 32 (anger) / 64 (speaker)
**Loss:** Binary cross-entropy (sigmoid output)
**Callbacks:** EarlyStopping (val_acc) + ReduceLROnPlateau (val_loss)
**Random seed:** 42

### Dataset: ESD (Emotional Speech Database)

Clasificación binaria: Angry vs. Other

Entrenamiento: 12,000 utterances

Test: 1,200 utterances (balanceado)

Evaluación: 800 utterances

MFCC: 20 bandas × 63 frames por utterance

### Post-Training Quantization

Per-tensor affine quantization (Jacob et al., 2018)

Set de calibración: 100 utterances representativas

**Resultado: .tflite INT8 de 28.8 KB**

# Referencia: Arquitectura LightCNN-9 Original (Facial)

*LightCNN-9 fue diseñada para reconocimiento facial con imágenes de 128×128 píxeles en escala de grises.*

| Capa | Operación | Salida | Activación |
|---|---|---|---|
| Input | Imagen gris | 128×128×1 | — |
| Conv1+Pool1 | 5×5, 96→48 + Pool | 64×64×48 | MFM |
| Conv2a+Conv2+Pool2 | 1×1+3×3, 192→96 + Pool | 32×32×96 | MFM |
| Conv3a+Conv3+Pool3 | 1×1+3×3, 384→192 + Pool | 16×16×192 | MFM |
| Conv4a+Conv4+Pool4 | 1×1+3×3, 256→128 + Pool | 8×8×128 | MFM |
| Conv5a+Conv5+Pool5 | 1×1+3×3, 256→128 + Pool | 4×4×128 | MFM |
| FC1+Output | 512→256 + N clases | N | MFM+Softmax |

MicroLightCNN preserva los kernels 3×3 y el max-pooling agresivo, pero reduce a 3 bloques conv, reemplaza MFM por ReLU, y opera sobre MFCC 20×63 (voz) en lugar de imágenes 128×128 (rostro).

# MFM vs ReLU: Justificación del Cambio

## MFM (LightCNN original)

**Operación:** $\max(x_1, x_2)$
**Entrada:** 2 mapas de características
**Salida:** Mitad de canales
**Mecanismo:** Competencia entre filtros
*Requiere Split + Maximum (ausentes en TFLite-M)*
Ideal para multiclase con etiquetas ruidosas

## ReLU (MicroLightCNN)

**Operación:** max(0, x)
**Entrada:** 1 mapa de características
**Salida:** Misma dimensión
**Mecanismo:** Umbral simple (sparsity by zeroing)
***Soportado nativamente en TFLite-M***
Estable bajo SNR-based noise injection

Para clasificación binaria, la inhibición competitiva de MFM agrega poco. ReLU es suficiente y permite el despliegue en MCU.

# Conclusiones y Contribuciones

1

## MicroLightCNN

Mayor accuracy INT8 (83.2%) entre detectores de enojo desplegables en MCU, con el menor arena (60 KB).

2

## Pipeline Dual

Primer sistema embebido que detecta emoción e identifica hablante simultáneamente en un solo MCU.

3

## Cuantización Beneficiosa

INT8 no solo preserva sino que MEJORA accuracy (+0.4 pp) gracias al efecto regularizador.

4

## Evaluación Rigurosa

Comparación head-to-head contra DS-CNN, MatchboxNet, ECAPA-TDNN y CAM++ bajo presupuesto <50k params.

# Gracias

Dr. Hugo Arnoldo Mitre Hernández

¿Preguntas?