

MARÍA MARTA RAMIREZ

DIEGO ALBERTO LEIVA

JOSÉ PABLO ORELLANA

# CNN

## URBAN SOUND CLASSIFICATION



# Descripción del Problema



El entorno urbano está repleto de una variedad de sonidos que van desde ruidos de tráfico y obras de construcción hasta sonidos de la naturaleza y actividades humanas. La clasificación automática de estos sonidos es un campo de investigación emergente con aplicaciones en la recuperación de multimedia y en la mejora de la calidad de vida en las ciudades.

# Análisis

La clasificación de sonidos ambientales implica múltiples retos técnicos.

## DIVERSIDAD Y COMPLEJIDAD

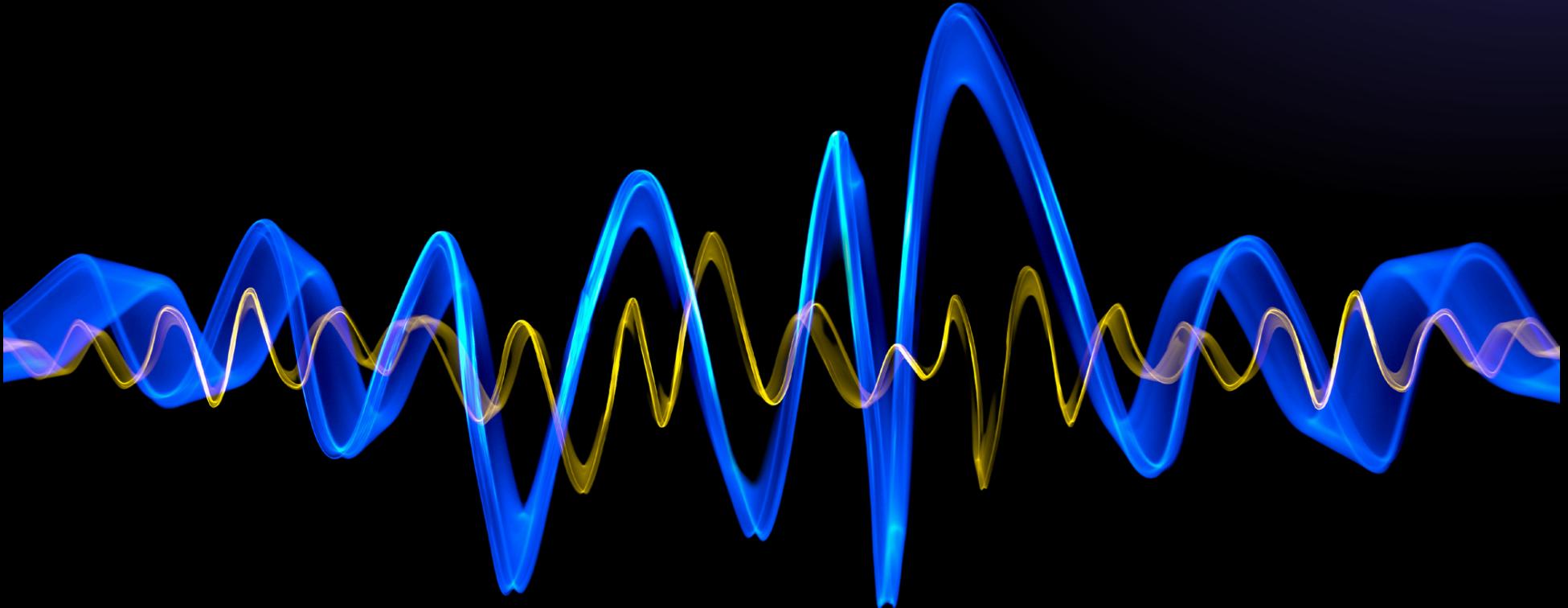
Los sonidos urbanos son diversos y complejos; cambian en intensidad y pueden solaparse, dificultando su clasificación.

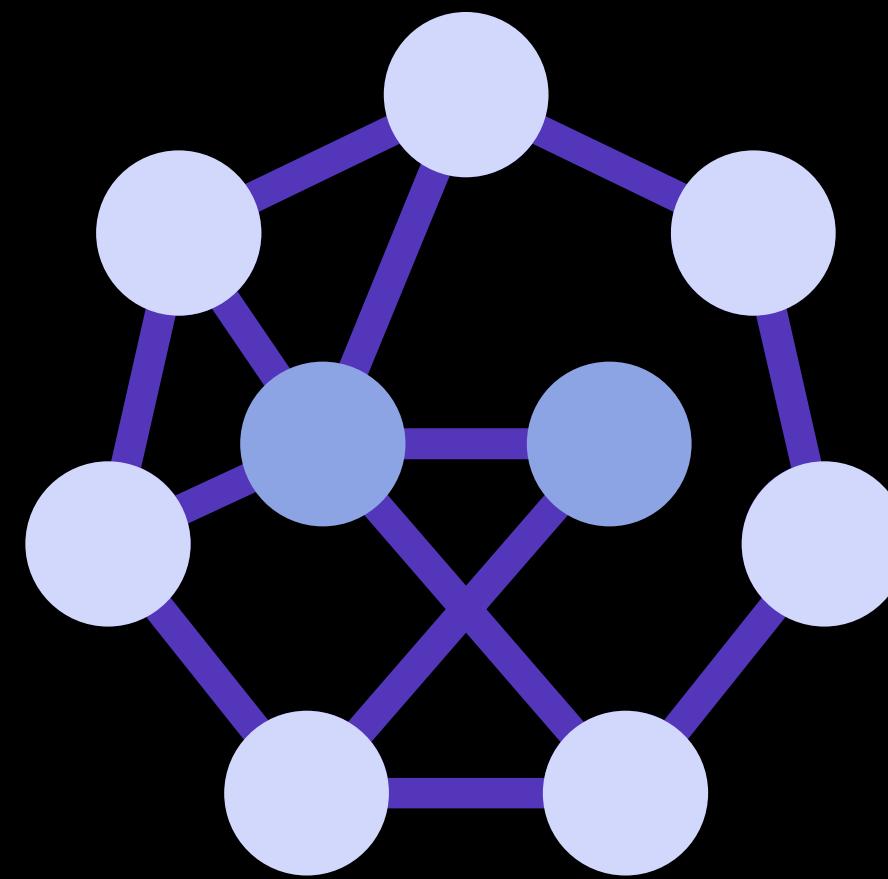
## ACÚSTICA URBANA

El caos acústico urbano hace que diferentes fuentes de sonido se mezclen, complicando la identificación precisa de cada fuente.

## DATASET URBANSOUND

UrbanSound ofrece una taxonomía con 10 clases de sonidos urbanos etiquetados, permitiendo entrenar modelos de clasificación más efectivos.





## Propuesta de la Solución

Para abordar el problema de clasificación de sonidos urbanos, proponemos implementar un modelo de redes neuronales convolucionales (CNN) que tome como entrada representaciones espectrales de los sonidos del dataset UrbanSound.

### 01 Preprocesamiento de los datos

Los audios fueron convertidos en spectrogramas, que son representaciones visuales de las frecuencias a lo largo del tiempo.

### 03 Entrenamiento y Evaluación

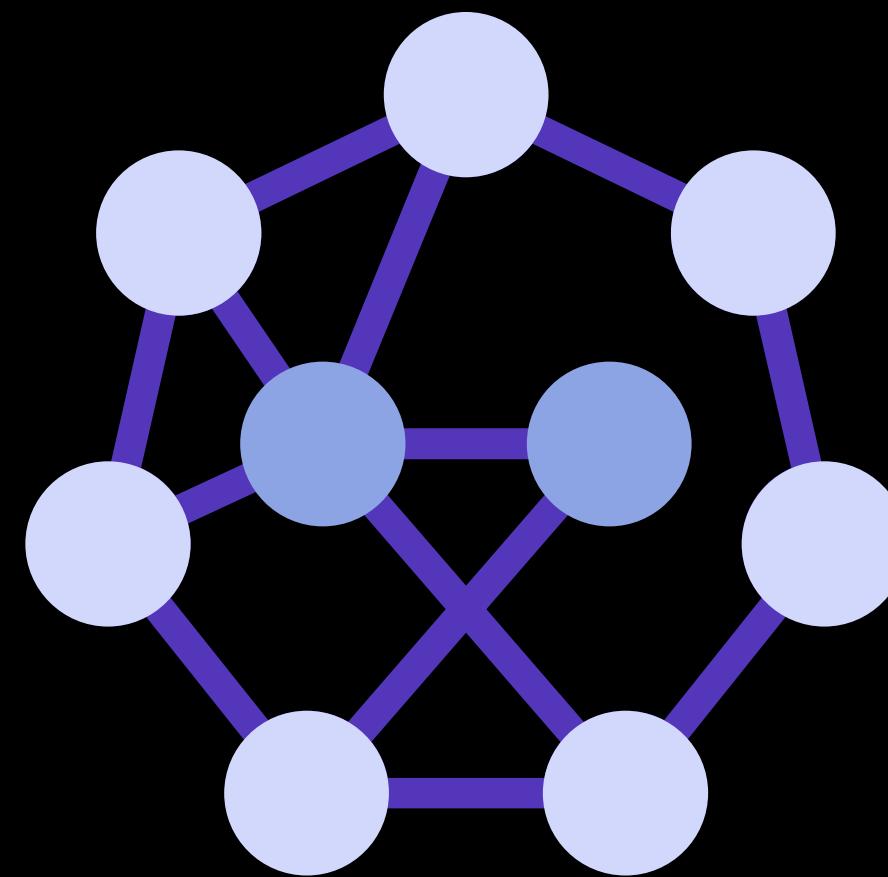
Se entra el modelo en el dataset UrbanSound, ajustando hiperparámetros clave y aplicando técnicas como la regularización y el aumento de datos para mejorar la generalización del modelo.

### 02 Diseño e Implementación del Modelo

Desarrollaremos una arquitectura de red CNN optimizada para el análisis de spectrogramas. Esto implica varias capas de convolución y pooling.

### 04 Optimización y Mejora

A medida que avanzamos en el entrenamiento, consideraremos otras técnicas avanzadas de Deep Learning, como la transferencia de aprendizaje y el ajuste fino.



## Propuesta de la Solución

Otra de las soluciones que se consideraron implementar es el uso de modelos basado en redes neuronales recurrentes de tipo LSTM, estas redes son ideales para procesar secuencias temporales, como lo podría ser una señal de audio. Todo gracias a su capacidad para capturar patrones temporales complejos y relaciones a largo plazo.

### 01 Preprocesamiento de los datos

- Las señales de audio serán transformadas en espectrogramas de Mel y coeficientes MFCC, aplicando técnicas de normalización y enmascaramiento (de tiempo y frecuencia) para robustecer el modelo ante variaciones en los datos.

### 03 Entrenamiento y Evaluación

- El modelo será entrenado en el conjunto de datos UrbanSound, ajustando parámetros como el número de capas, tamaño de lote y tasa de aprendizaje.

### 02 Diseño e Implementación del Modelo

Se desarrollará una arquitectura de LSTM bidireccional con varias capas ocultas para procesar secuencias temporales derivadas de los espectrogramas de Mel o MFCC.

### 04 Optimización y Mejora

- Se evaluará el desempeño del modelo utilizando métricas como precisión, sensibilidad y F1-score, con un análisis específico de su efectividad en clases minoritarias.

# Herramientas Aplicadas

El lenguaje de programación utilizado para el desarrollo de este proyecto fue Python, seleccionado por su amplia variedad de bibliotecas especializadas en Deep Learning y procesamiento de audio, como PyTorch, que facilitaron la implementación del modelo y el preprocesamiento de datos.

## Librerías y Frameworks

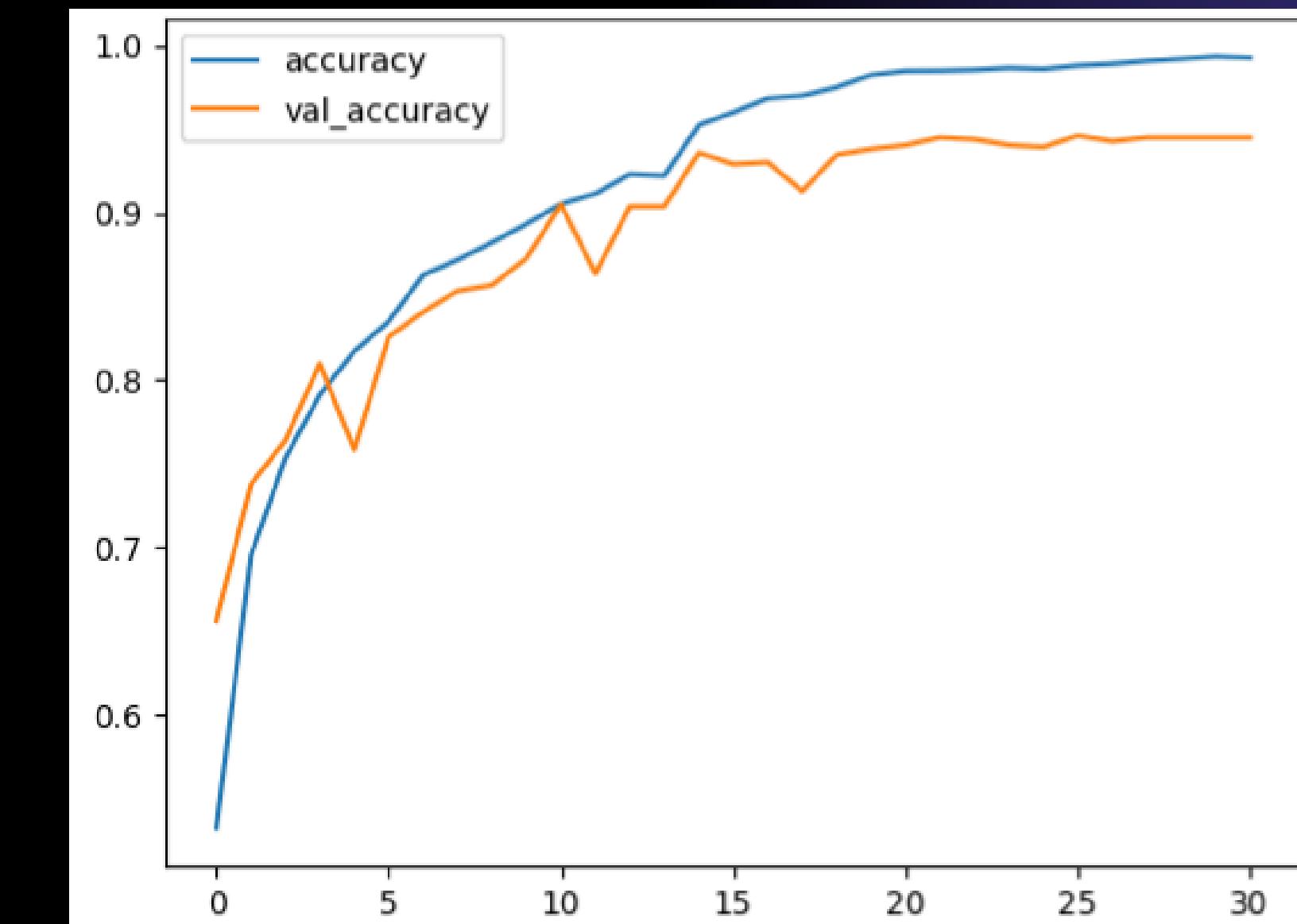
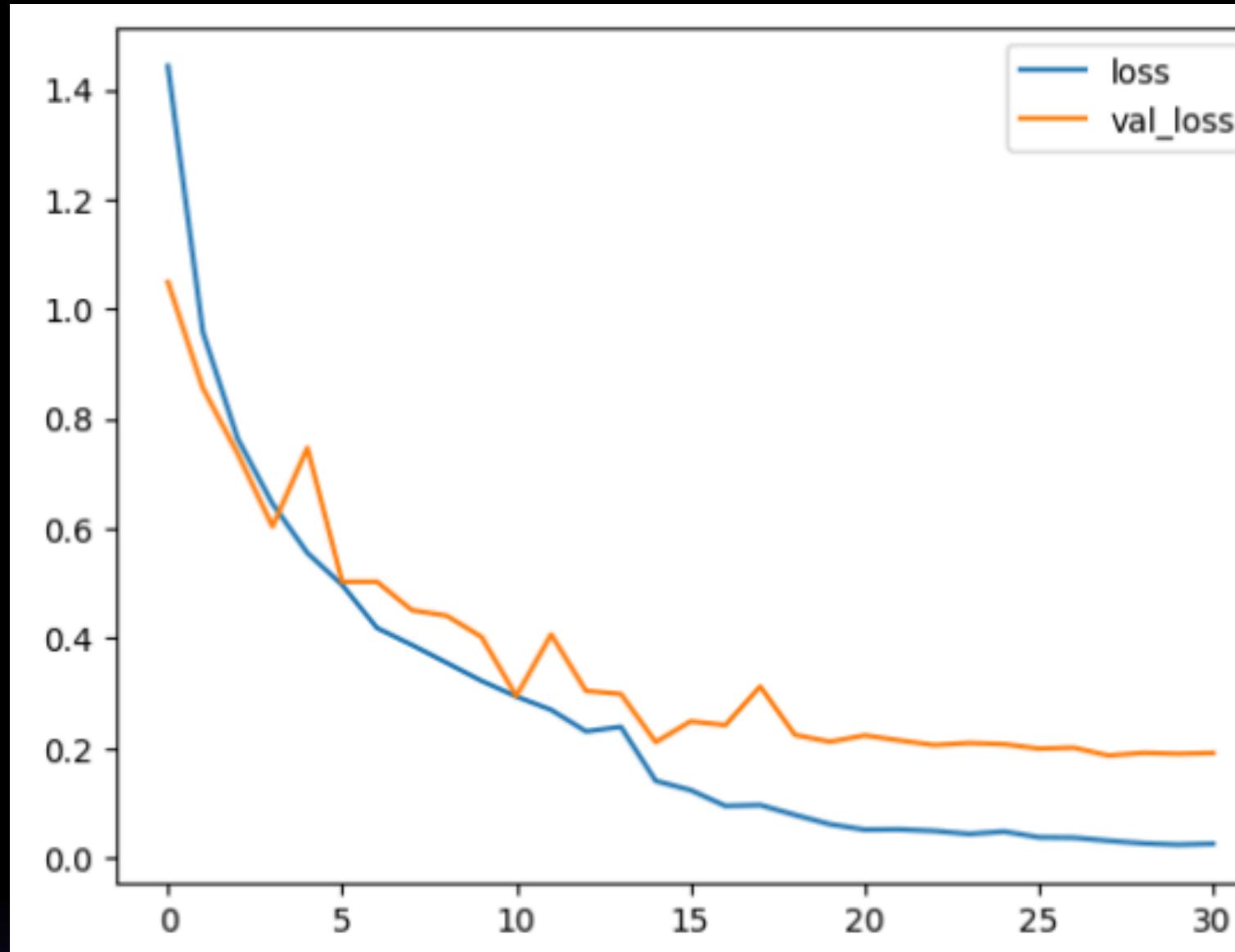
- **Pandas:** Para manejar y procesar los metadatos del dataset UrbanSound8K.
- **NumPy:** Para realizar cálculos y manipular estructuras de datos numéricas como arrays.
- **Matplotlib y Seaborn:** Para crear visualizaciones, como curvas de precisión y pérdida, matrices de confusión y distribuciones de datos.
- **Scikit-learn:** Métricas como classification report y confusion matrix para evaluar el rendimiento del modelo.
- **PyTorch:** Para construir y entrenar el modelo LSTM.
- **Torchaudio:** Para el procesamiento de señales de audio, como la conversión a espectrogramas de Mel y la normalización.
- **CUDA:** Para acelerar el entrenamiento del modelo utilizando GPU.
- **Keras (TensorFlow):** Para construir y entrenar los modelos CNN.
- **Librosa:** Utilizado en ambos modelos para el procesamiento de audio, como la carga de señales de audio, generación de espectrogramas de Mel, extracción de coeficientes MFCC y augmentación de datos.
- **TQDM:** Para visualizar barras de progreso durante la ejecución de tareas largas.



# Resultados

Modelo CNN con spectrograma y MFCCs

Este nuevo modelo que introduce el uso de MFCCs (Coeficientes Cepstrales en las Frecuencias de Mel) obtuvo un incremento significativo en rendimiento llegando a un 94.62% de accuracy tras 50 épocas de entrenamiento.



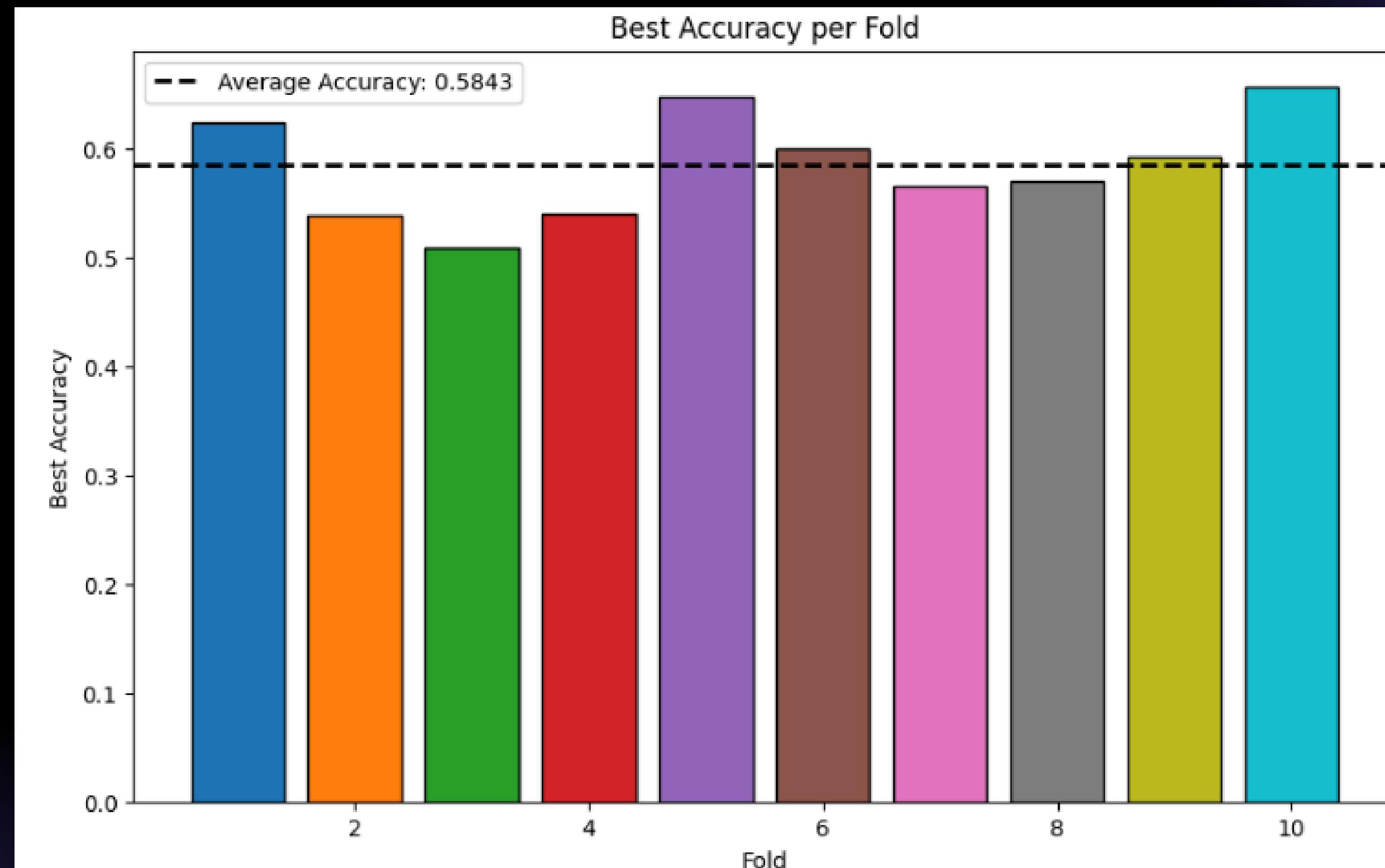
MODELO CNN CON ESPECTROGRAMA LINEAL  
ESTE PRIMER MODELO OBTUVO UN ACCURACY GENERAL DEL 54.53% TRAS 20 ÉPOCAS DE ENTRENAMIENTO

# Resultados

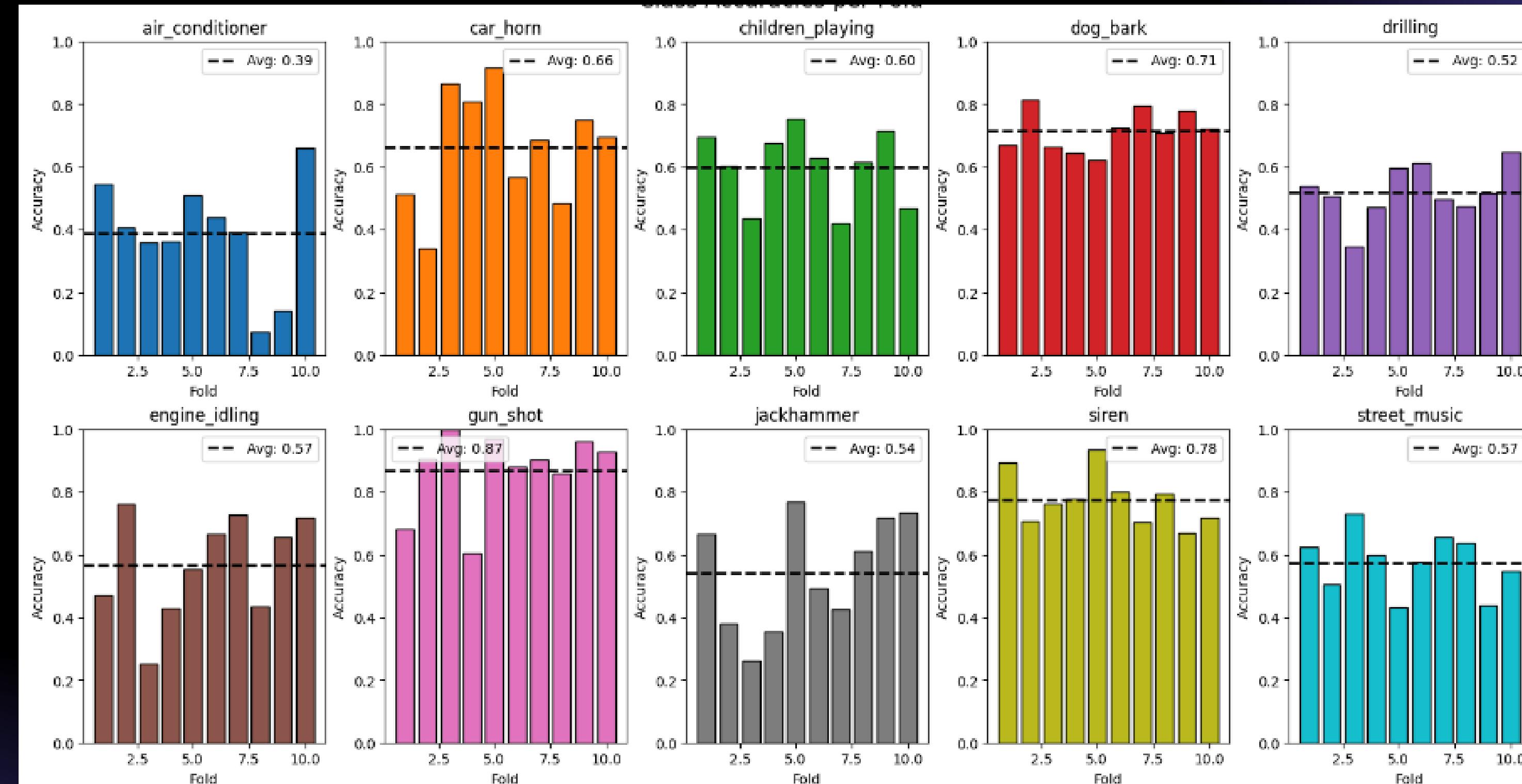
Modelo LSTM con espectrograma de mel y mecanismo de atención

Finalmente el modelo utilizado usó mecanismos de regularización como dropout, schedulers y mecanismo de atención para mejorar su rendimiento.

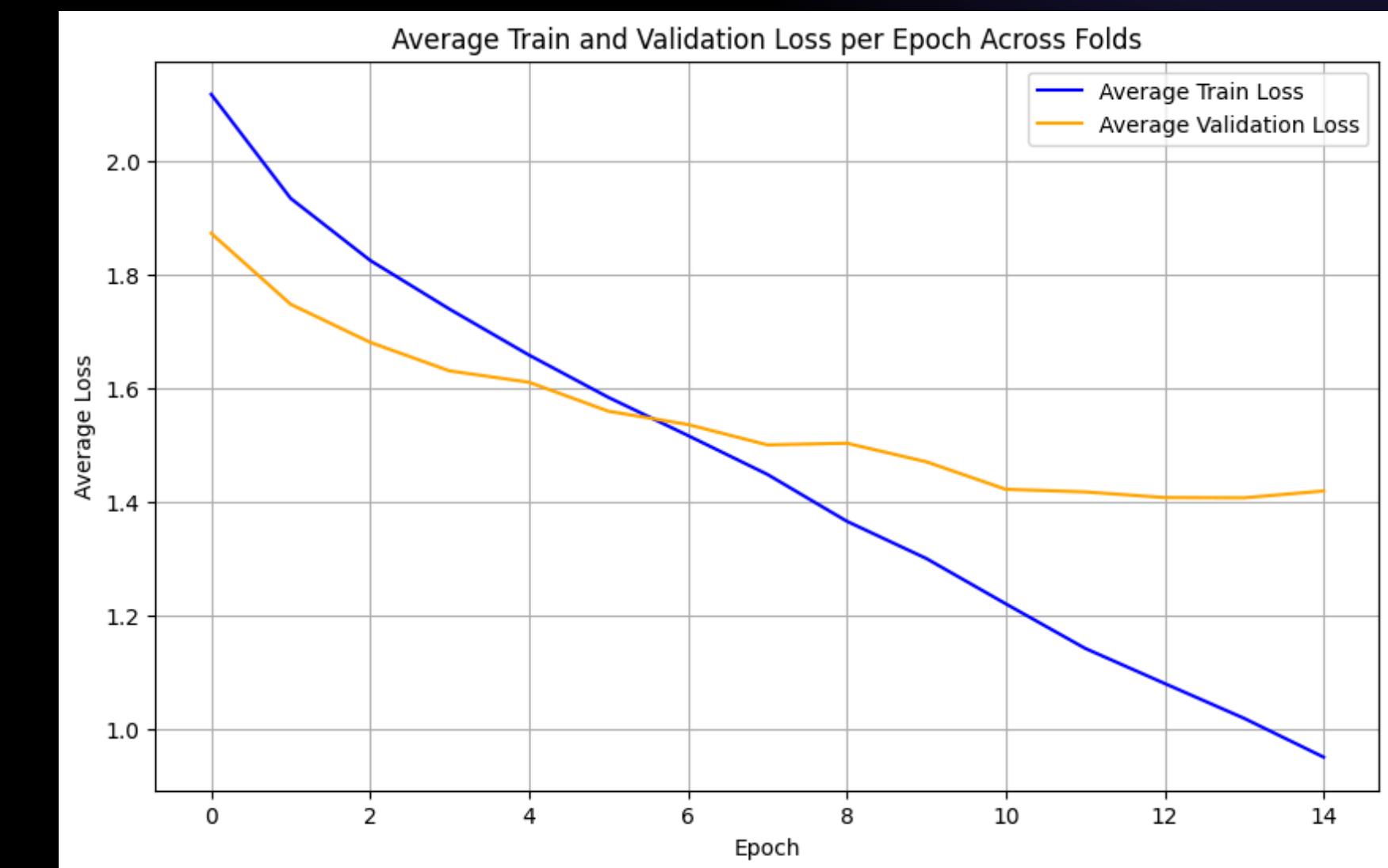
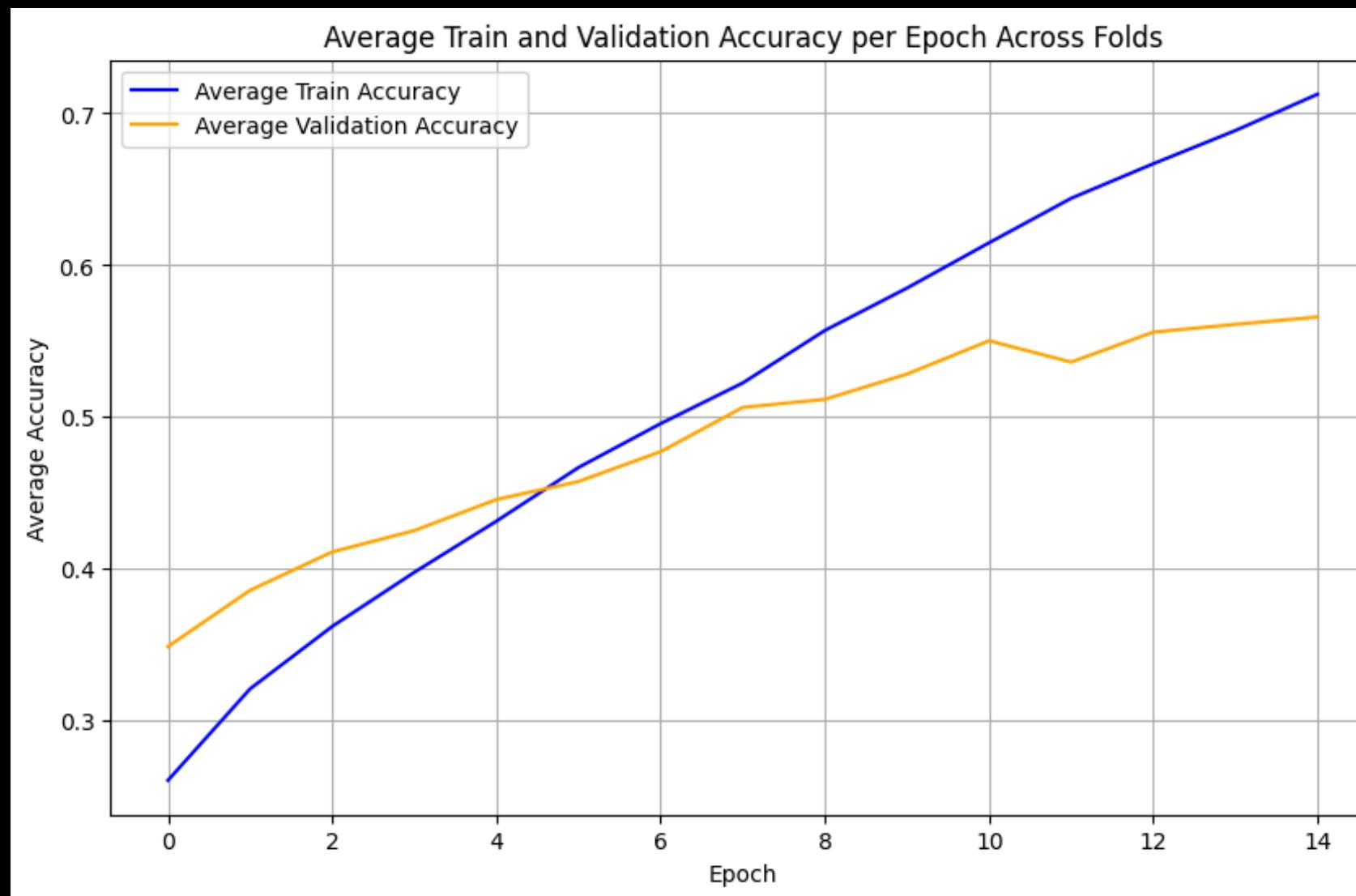
Nuevamente se aplicó una validación cruzada de 10 Folds para obtener el rendimiento general del modelo.



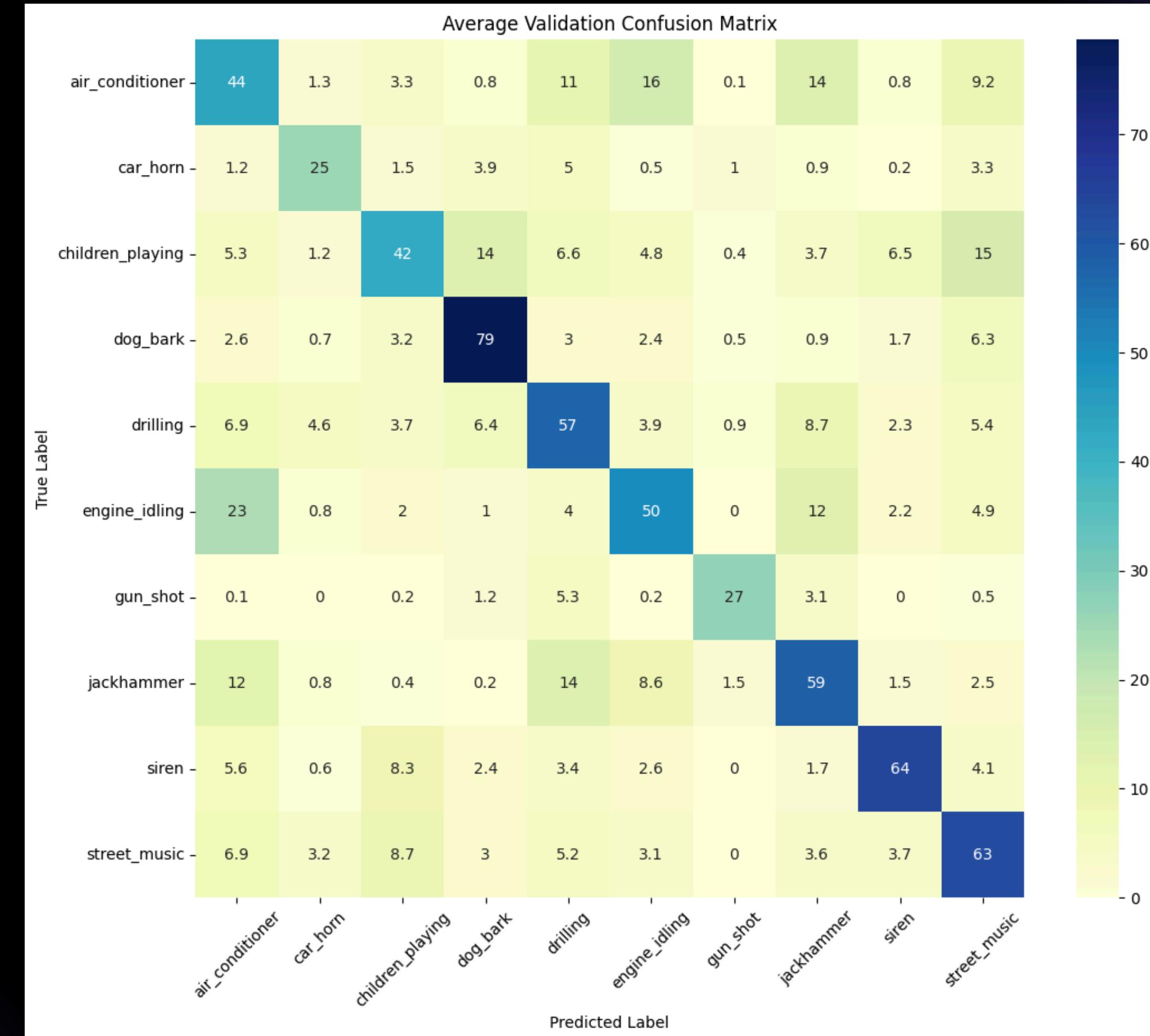
# Resultados



# Resultados



# Resultados



# Conclusión

El modelo mostró buen desempeño, especialmente al clasificar sonidos distintivos como sirena y dog\_bark, gracias al uso de características MEL. Sin embargo, tuvo dificultades con sonidos similares como aire acondicionado y motor. A pesar de esto, las métricas generales y la matriz de confusión evidencian su capacidad para diferenciar clases de manera efectiva en la mayoría de los casos.

# Muchas Gracias

MARÍA MARTA RAMIREZ

DIEGO ALBERTO LEIVA

JOSÉ PABLO ORELLANA