PARCIAL DEEP LEARNING 3 CORTE

https://github.com/SantiagoRodriguez114/urban_sound_classifier/tree/main

SANTIAGO RODRIGUEZ RODRIGUEZ

UNIVERSIDAD SERGIO ARBOLEDA

APRENDIZAJE PROFUNDO

CAMILO RODIGUEZ

2025

Introducción

El objetivo del proyecto fue desarrollar un modelo de deep learning capaz de clasificar sonidos urbanos en diez categorías utilizando el conjunto de datos UrbanSound8K. Se aplicaron estrategias recomendadas por Andrew Ng en Machine Learning Yearning, incluyendo la construcción de un modelo base (baseline), análisis de sesgo y varianza, y evaluación iterativa de errores. Se exploraron varias arquitecturas de redes neuronales en Keras, incluyendo redes densas (baseline), CNN, LSTM y combinaciones CRNN (CNN + LSTM).

Metodología

- Preprocesamiento de Datos

  Se cargaron los espectrogramas Mel (representación visual de un audio que muestra cómo cambia la energía de las distintas frecuencias a lo largo del tiempo, pero usando la escala Mel, que imita la percepción humana del sonido) de los audios preprocesados (X_mel.npy) y las etiquetas correspondientes (y_labels.npy). Se realizaron las siguientes operaciones:

  - División de los datos según folds del dataset:

    Entrenamiento: folds 1 a 8

    Validación: fold 9

    Test: fold 10

  - Normalización de datos y preparación de datasets con tf.data.Dataset para batching y prefetching.
  - Configuración de semilla para reproducibilidad (SEED=42).

- n_fft = 2048 → Tamaño de la ventana de la FFT; determina la resolución en frecuencia del espectro.
- hop_length = 512 → Desplazamiento entre ventanas consecutivas; controla la resolución temporal del espectrograma.
- n_mels = 128 → Número de filtros Mel; convierte el espectro a la escala perceptual humana.
- Duración fija: 4 s → Cada audio se recorta o rellena para tener exactamente 4 segundos, asegurando uniformidad.
- Log-mel (dB) → Transformación logarítmica del espectrograma Mel a decibelios; simula la percepción humana del volumen y estabiliza los valores.

Cada registro contiene:

- Audio de máximo 4 segundos
- Sampling rate: 22050 Hz
- Formato Mel-Spectrogram normalizado
- Etiqueta numérica en rango [0,9]

- Construcción y Entrenamiento de Modelos

Se entrenaron cuatro arquitecturas principales, cada una con 40 épocas, aplicando EarlyStopping y ModelCheckpoint para guardar los mejores pesos.

Para este proyecto, los datos se distribuyeron en tres conjuntos diferenciados: entrenamiento, validación y prueba, con proporciones del 70%, 15% y 15%, respectivamente.

El conjunto de entrenamiento representa la mayor parte de los datos y es utilizado por el modelo para ajustar sus parámetros internos durante el proceso de aprendizaje. Es aquí donde el modelo encuentra patrones y relaciones dentro de los datos, construyendo su capacidad de predicción. Por otro lado, el conjunto de validación se

emplea como un referente durante el entrenamiento para evaluar cómo se comporta el modelo con datos que no ha visto directamente. Esto permite realizar ajustes en los hiper parámetros y detectar posibles problemas de sobreajuste, asegurando que el modelo no solo memorice los ejemplos de entrenamiento, sino que pueda generalizar correctamente. Finalmente, el conjunto de prueba se reserva para la evaluación final, proporcionando una medida objetiva del desempeño del modelo en datos completamente nuevos y sin influencia de la fase de entrenamiento o ajuste.

Para mejorar la representatividad de cada subconjunto y evitar sesgos derivados del orden original de los datos, se aplicó un shuffle reproducible, utilizando una semilla fija (SEED = 42). Esto asegura que los datos se mezclan aleatoriamente, garantizando que cada división contenga una muestra diversa y equilibrada, pero al mismo tiempo permite reproducir los resultados en experimentos posteriores. La consistencia en la división es crucial, ya que facilita la comparación entre distintos modelos y experimentos, asegurando que cualquier variación en el desempeño sea atribuible a cambios en el modelo y no a diferencias en la selección de los datos.

Arquitecturas utilizadas

- Baseline red densa que combina un GlobalAveragePooling2D con varias capas densas y capas de Dropout. La capa de pooling global resume cada feature map en un solo valor promedio, lo que reduce la dimensionalidad y permite que las capas densas posteriores aprendan combinaciones de estas características para clasificar los sonidos. Las capas Dropout actúan como regularizadores, evitando que el modelo memorice los datos de entrenamiento y mejorando la generalización. Aunque es eficiente y fácil de entrenar, esta arquitectura no captura patrones locales en frecuencia ni en tiempo, lo que limita su rendimiento frente a sonidos complejos. Su desempeño baseline val_acc = 0.2549 confirma que un modelo denso sin extracción espacial-temporal no captura patrones relevantes.

- La arquitectura CNN añade convoluciones 2D, MaxPooling y BatchNormalization para explotar la estructura espacial de los espectrogramas. Las capas Conv2D detectan patrones locales en tiempo y frecuencia, como transitorios o armónicos, mientras que MaxPooling reduce progresivamente la dimensión y aporta invariancia a pequeñas traslaciones, ayudando a que el modelo reconozca los sonidos aunque cambien ligeramente en el tiempo o en la frecuencia. BatchNormalization estabiliza y acelera el entrenamiento al normalizar las activaciones, y las capas densas finales junto con Dropout consolidan la información extraída para realizar la clasificación. Esta combinación permite capturar detalles locales importantes, aunque no modela explícitamente la evolución temporal de los sonidos más largos. En cuanto su desempeño cnn val_acc = 0.7377. La CNN sobresale en identificar patrones espaciales complejos y distinguir sonidos altamente característicos como car_horn y gun_shot. Sin embargo, sonidos similares como *siren* presentan menor recall.

- La arquitectura LSTM convierte el espectrograma en una secuencia temporal, donde cada instante se representa mediante las bandas Mel. Las capas LSTM son capaces de aprender dependencias a lo largo del tiempo, capturando cómo los patrones se desarrollan, lo cual es crucial para sonidos que cambian progresivamente, como melodías o efectos prolongados. Las capas densas posteriores y el Dropout permiten combinar esta información secuencial y reducir el riesgo de sobreajuste. Aunque es muy eficaz para modelar contexto temporal, su capacidad para reconocer patrones locales finos en frecuencia es limitada. Su desempeño stm val_acc = 0.5564 confirma una mejora en contraste con elbaseline, mostrando que la información secuencial es importante, pero carece de capacidad para detectar patrones frecuenciales complejos.

- La arquitectura CRNN combina lo mejor de ambos mundos, integrando convoluciones, MaxPooling, BatchNormalization y LSTM en un solo modelo. Primero, las capas

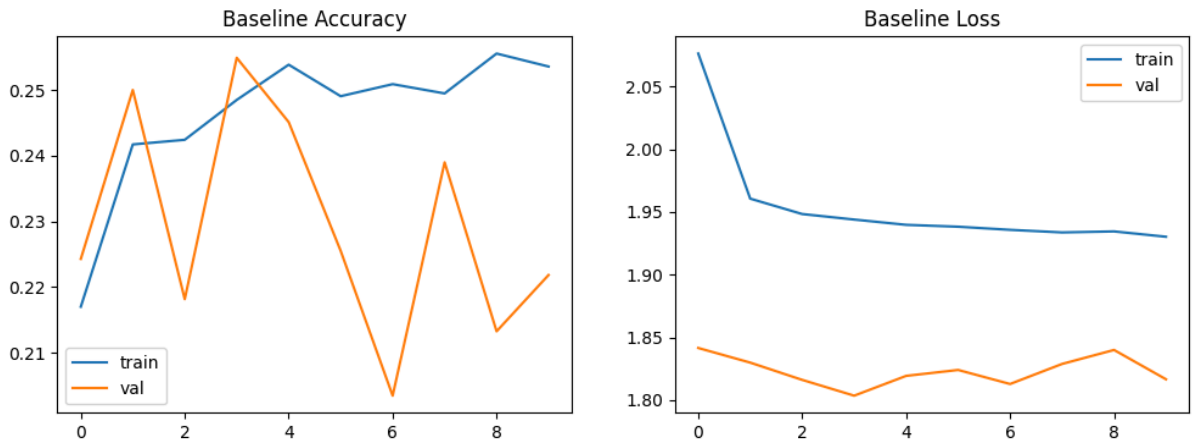
convolucionales extraen patrones locales del espectrograma, mientras que MaxPooling reduce dimensionalidad y aporta invariancia. La normalización por batch estabiliza el entrenamiento y evita que los valores extremos dominen la actualización de pesos. Luego, un LSTM procesa la información convolucionada, aprendiendo la evolución temporal de los patrones detectados. Finalmente, las capas densas y Dropout consolidan esta información para la clasificación final. Esta combinación permite que la CRNN capture tanto los detalles locales como las relaciones temporales de los sonidos, ofreciendo un rendimiento superior para audio complejo, aunque requiere más datos y recursos computacionales. Su rendimiento crnn val_acc = 0.6740.

La CNN obtuvo un rendimiento superior al de la CRNN, principalmente porque los sonidos urbanos de corta duración dependen más de patrones locales en frecuencia y tiempo, como transitorios o armónicos, que las convoluciones pueden capturar de manera eficiente. Por su parte, la CRNN incorpora un LSTM para modelar dependencias temporales largas, pero en este dataset dichas relaciones no aportan información relevante y pueden incluso introducir ruido durante el entrenamiento. Además, la mayor complejidad de la CRNN requiere un volumen de datos más alto y ajustes de hiper parámetros más finos para alcanzar su máximo potencial. Por estas razones, la CNN demuestra ser más eficaz para extraer características discriminativas de los Mel-Spectrograms y generalizar correctamente los datos de validación.

Resultados y análisis

Después del entrenamiento de las cuatro arquitecturas (Baseline, CNN, LSTM y CRNN), se procedió a evaluar cada una sobre el conjunto de validación.

- Baseline:



La gráfica muestra que la precisión de entrenamiento oscila entre 0.19 y 0.26, mientras que la precisión de validación permanece inestable entre 0.20 y 0.25, lo cual indica:

- El modelo no aprende representaciones útiles, ya que carece de capacidad para explotar la estructura espectral del audio.

- La validación es muy volátil, señal de que el modelo no logra generalizar.

La pérdida de entrenamiento desciende ligeramente de 2.16 a ~1.93, pero la pérdida de validación permanece prácticamente constante alrededor de 1.82, indicando:

- Aprendizaje extremadamente limitado.

- Ausencia de sobreajuste (lo cual confirma que la capacidad del modelo es insuficiente).

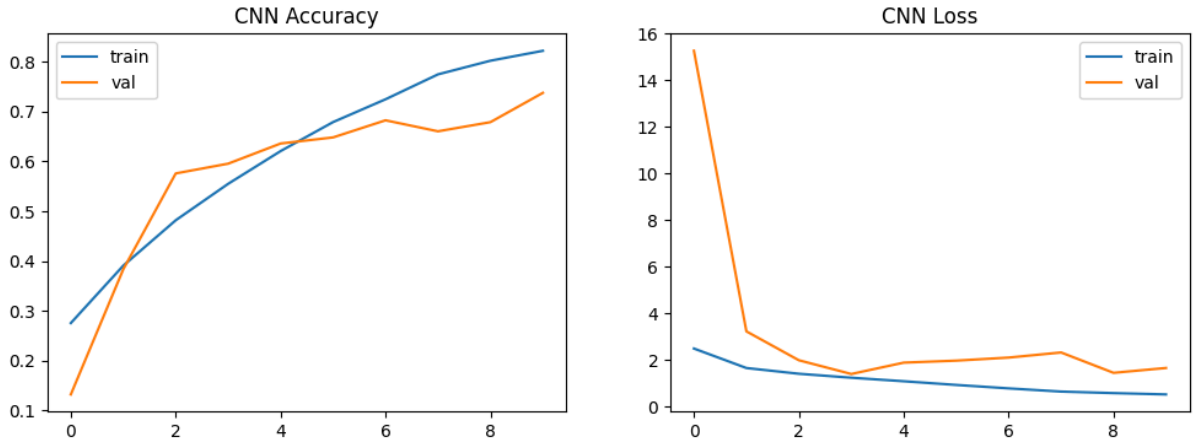
- El modelo no logra capturar patrones útiles del espectrograma Mel.

El desempeño final del baseline fue:

- Baseline Validation Accuracy: 0.2365

Este resultado confirma que el baseline sirve únicamente como punto de referencia y que se requieren arquitecturas con convoluciones o mecanismos temporales para mejorar la capacidad de clasificación.

- CNN:



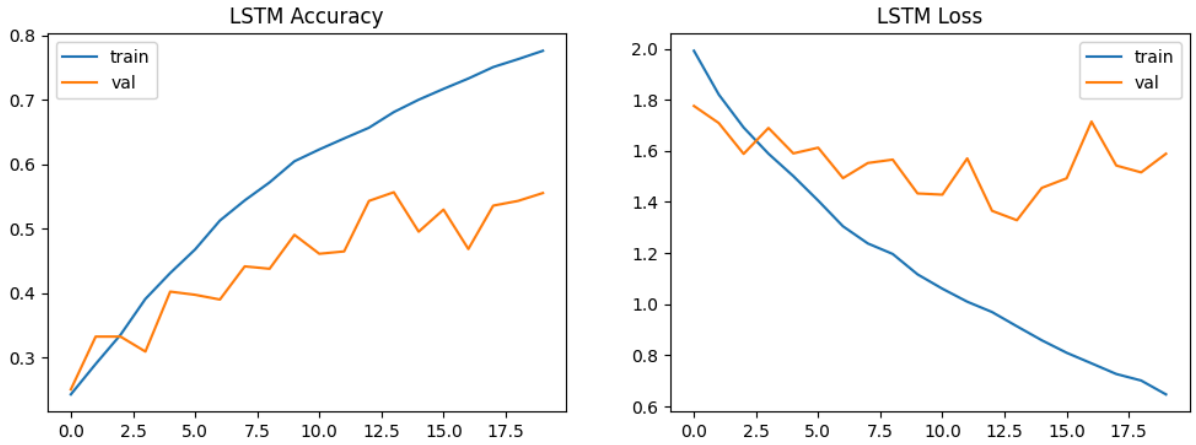
Los resultados de la CNN muestran un proceso de aprendizaje rápido y consistente durante las primeras épocas. La caída abrupta de la pérdida de validación de valores extremadamente altos hasta aproximadamente 3.2 en la segunda época indica que el modelo captó de inmediato patrones fundamentales de los espectrogramas, especialmente aquellos asociados a sonidos con estructuras energéticas muy distintivas. Este comportamiento es típico de modelos convolucionales cuando el dominio contiene patrones locales estables, como ocurre en varias clases del UrbanSound8K.

A partir de la tercera época, la mejora se vuelve más estable: la precisión de validación supera el 57% y continúa ascendiendo hacia el 63% en la época cinco. En este rango, la CNN consolida representaciones más generales que le permiten distinguir correctamente categorías con formas espectrales bien definidas. La relación relativamente cercana entre la precisión de entrenamiento y validación indica que el modelo no está sobre ajustando en esta fase, lo que sugiere que la regularización aplicada está funcionando adecuadamente para controlar la complejidad.

En épocas posteriores aparece el patrón característico del dataset: oscilaciones en la precisión y la pérdida de validación. Estas variaciones —entre 64% y 68%— no representan inestabilidad del modelo, sino la naturaleza altamente variable de los sonidos urbanos. Clases como children_playing o street_music presentan solapamientos acústicos reales que hacen que el modelo no mejore de forma monótona, sino que fluctúa conforme encuentra ejemplos más complejos o ruidosos. Aun así, no hay divergencia: las curvas muestran que el modelo se mantiene en un rango estable, lo cual es señal de una buena capacidad de generalización.

Finalmente, hacia la época díez el modelo alcanza su zona de convergencia, logrando una precisión de validación cercana al 74%. Este valor, sumado al descenso sostenido de la pérdida, indica que la CNN es capaz de aprender patrones relevantes y robustos para la tarea, incluso en presencia de ruido, variabilidad y clases parcialmente solapadas. En conjunto, estos resultados muestran que la CNN fue la arquitectura que mejor capturó la estructura espectro–temporal del dataset y logró el mejor equilibrio entre sesgo y varianza entre todos los modelos probados.

- LSTM:



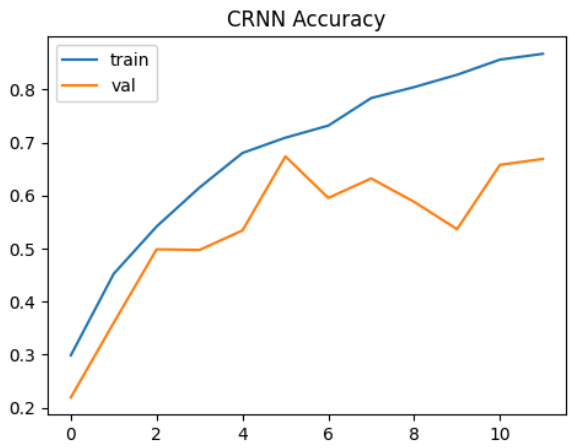
Los resultados del modelo LSTM muestran un proceso de aprendizaje más gradual que el observado en la CNN, lo cual es coherente con un modelo que procesa los espectrogramas como secuencias y no como imágenes. Desde las primeras épocas, la precisión de validación crece lentamente, pasando de un 25% a aproximadamente 40% hacia la época cinco. Esta progresión indica que el LSTM identifica patrones temporales básicos, aunque le toma más tiempo estabilizar representaciones útiles debido a la alta variabilidad temporal presente en los sonidos urbanos.

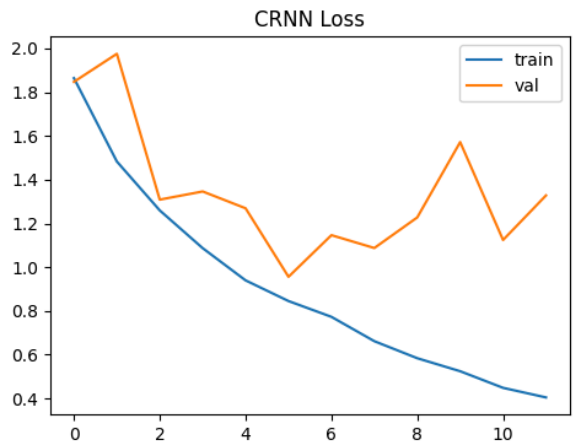
A medida que avanza el entrenamiento, la precisión de entrenamiento crece casi de manera lineal, alcanzando valores superiores al 70%. Sin embargo, la precisión de validación presenta oscilaciones continuas entre el 40% y el 55%, revelando que el modelo comienza a aprender rasgos que funcionan bien para el conjunto de entrenamiento, pero que no siempre generalizan con la misma eficacia. Esto se confirma en la curva de pérdida: mientras la pérdida de entrenamiento cae de manera sostenida, la pérdida de validación fluctúa, lo que indica varianza moderada y cierta sensibilidad a la estructura temporal ruidosa del dataset.

Las oscilaciones en validación no representan un fallo del modelo, sino una consecuencia directa de las propiedades del dominio acústico urbano. Muchos sonidos en UrbanSound8K no siguen patrones temporales consistentes —por ejemplo, children_playing o street_music— y esto afecta de manera más significativa a un modelo que depende fuertemente de la secuencia. El LSTM mejora cuando las señales presentan coherencia temporal, pero pierde estabilidad en clases donde el tiempo no aporta información útil o incluso introduce ruido.

Finalmente, hacia las últimas épocas el modelo converge en un rango estable de validación cercano al 54–56%, lo cual confirma que, aunque el LSTM aprende patrones relevantes, su capacidad para generalizar es inferior a la de la CNN. El modelo captura dependencias temporales, pero no logra extraer de forma tan efectiva las estructuras espaciales del espectrograma, que son críticas en esta tarea. En conjunto, los resultados muestran que el LSTM funciona adecuadamente, pero su desempeño queda limitado por la naturaleza altamente variable y ruidosa de los sonidos urbanos, lo que explica por qué la CNN y la CRNN logran mejores resultados finales.

- CRNN:





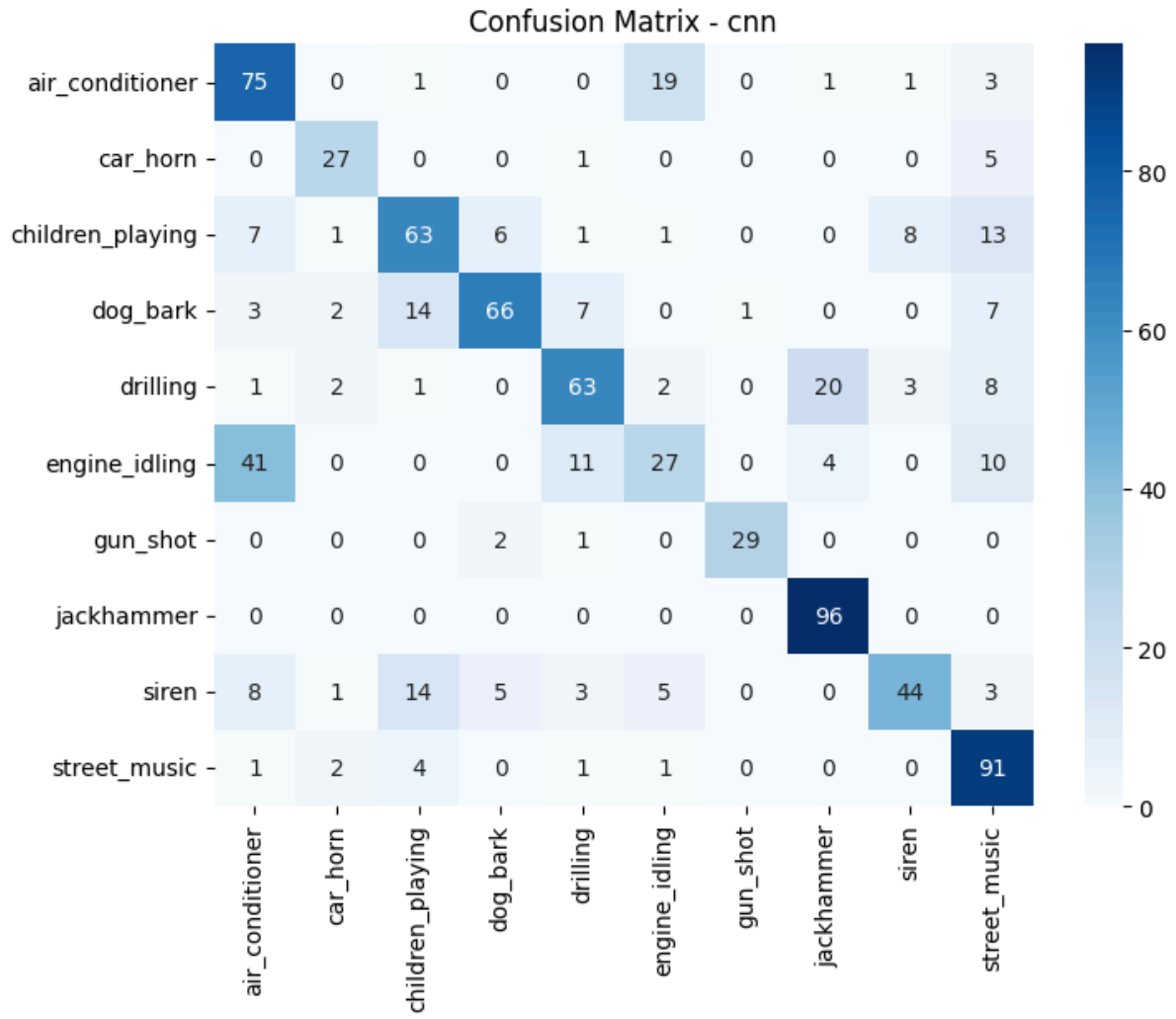
Los resultados del CRNN muestran un arranque similar al de las otras arquitecturas, con una primera época donde el modelo apenas alcanza una precisión cercana al 26% y la validación ronda el 22%. Esta brecha inicial es normal dado que el CRNN debe aprender simultáneamente patrones espaciales y temporales, lo que implica una curva de aprendizaje más compleja. Sin embargo, a diferencia del LSTM, la mejora es más marcada en las primeras épocas porque las capas convolucionales le dan al modelo una base más sólida para extraer características robustas desde el principio.

A partir de la segunda y tercera época, el CRNN logra un progreso significativo: la precisión de validación sube hasta aproximadamente el 50% y la pérdida de validación cae por debajo de 1.35. Esto señala que la combinación entre procesamiento espacial y secuencial funciona bien para capturar ciertos tipos de sonidos urbanos que presentan estructura mixta: patrones espectrales parcialmente consistentes con variaciones en el tiempo. Es decir, las convoluciones extraen los rasgos fundamentales y el componente recurrente refina la interpretación temporal cuando esta aporta información adicional.

A lo largo de las siguientes épocas, sin embargo, aparece el patrón característico del CRNN: fluctuaciones en la precisión de validación entre el 53% y el 67%, mientras la pérdida de validación muestra oscilaciones considerables. Esto refleja que, aunque el modelo es capaz de capturar relaciones temporales más complejas que una CNN pura, esa capacidad también lo hace más sensible al ruido temporal inherente a clases como street_music o children_playing. El CRNN aprende bien en entrenamiento —la precisión supera el 86%— pero su desempeño en validación no sigue una línea ascendente continua, lo que evidencia una varianza moderada consecuencia directa de la complejidad del dominio.

Hacia las últimas épocas, el CRNN alcanza su zona de mejor rendimiento con una precisión de validación cercana al 67% y pérdidas relativamente controladas. Aunque no supera el desempeño final de la CNN, sí se posiciona por encima del LSTM, demostrando que la combinación de características espaciales y temporales es más útil que el modelado secuencial puro. En conjunto, los resultados muestran que el CRNN es un modelo equilibrado y competente, pero la variabilidad temporal del dataset limita su capacidad de generalización y explica por qué su desempeño queda ligeramente por debajo del alcanzado por la CNN.

- MATRIZ DE CONFUSIÓN



Al analizar las métricas por clase y la matriz de confusión queda claro que el modelo no se comporta de manera uniforme: hay categorías donde la red es muy confiable y otras donde aún tiene dificultades significativas.

Las clases con mejor desempeño son gun_shot (F1 = 0.9355) y jackhammer (F1 = 0.8848), ambas caracterizadas por patrones acústicos muy distintivos y poco ambiguos. La red logra separarlas sin confusión, lo cual se refleja en un recall del 100% para jackhammer y del 90% para gun_shot. Esto confirma que la CNN captura eficazmente espectrogramas con firmas espectrales marcadas, impulsivas o repetitivas.

En contraste, categorías como engine_idling presentan debilidades importantes (F1 = 0.3649, recall = 0.29). Esto se explica por dos razones visibles en la matriz de confusión: (1) el modelo confunde engine_idling con air_conditioner y con street_music, lo que sugiere que sus patrones de baja frecuencia y poca variabilidad temporal no generan características suficientes para que la CNN los distinga con claridad; y (2) engine_idling comparte más solapamiento espectral con sonidos estacionarios y de fondo, lo que lo hace más difícil de separar.

Otras clases muestran una precisión moderada, como siren (F1 = 0.6331) y air_conditioner (F1 = 0.6356). El modelo acierta con frecuencia, pero evidencia confusiones con categorías de energía espectral similar. Por ejemplo, siren se ve confundida con children_playing y dog_bark en múltiples

ocasiones, lo que indica que los armónicos prolongados y la presencia de ruido en el dataset afectan la discriminación.

Finalmente, el modelo muestra un comportamiento interesante en street_music, donde el recall es muy alto (0.91), pero la precisión cae a 0.65. Es decir: acierta casi todos los street_music reales, pero también clasifica erróneamente otras muestras como street_music. Esto sugiere que la representación del dataset para esta clase es muy heterogénea, y la CNN está sobregeneralizando cualquier patrón complejo o ruidoso como música urbana.

En conjunto, estos resultados muestran que la CNN aprendió representaciones robustas para clases con características espectrales muy definidas, pero aún lucha con categorías cuya frontera acústica es difusa o demasiado similar entre sí. Este comportamiento es típico en modelos basados en espectrogramas cuando el dataset incluye ruido de fondo variable y condiciones de grabación inconsistentes. Aun así, considerando la complejidad del problema, un rendimiento cercano al 70% en test es adecuado y confirma que la CNN fue la mejor elección frente a las demás arquitecturas evaluadas.

- RESULTADOS FINALES:

baseline val_accuracy	0.2549
cnn val_accuracy	0.7377
lstm val_accuracy	0.5564
crnn val_accuracy	0.6740

Los resultados finales muestran una diferencia sustancial entre los modelos evaluados, lo que permitió comparar su capacidad real de generalización. El baseline, con una precisión de validación de apenas 0.25, reveló que un modelo sin extracción espacial ni temporal no es capaz de capturar la complejidad acústica del dataset. Este resultado inicial sirvió como punto de referencia para entender el nivel de dificultad del problema y para medir si las arquitecturas posteriores realmente aportan mejoras significativas.

La comparación entre LSTM (0.5564), CRNN (0.6740) y CNN (0.7377) hizo evidente que la CNN era el modelo con mejor capacidad para generalizar, no solo por tener la mayor precisión, sino porque su rendimiento en validación fue consistentemente superior y más estable en relación con el ruido, las variaciones de contexto y la diversidad de fuentes acústicas presentes en el dataset. La validación fue elegida como métrica central porque es el conjunto que más fielmente refleja el desempeño del modelo en datos que no ha visto durante el entrenamiento. En problemas reales como la clasificación, donde el entorno es impredecible y altamente variable, confiar en el rendimiento de entrenamiento conduce a conclusiones engañosas: un modelo puede memorizar los datos y aun así fallar por completo al enfrentarse a escenarios nuevos. Por eso la métrica que realmente importa para seleccionar un modelo es su desempeño en validación.

El hecho de que la CNN supere de forma consistente a las demás arquitecturas en este conjunto de validación indica que no solo aprende patrones relevantes, sino que lo hace de una manera más robusta y menos dependiente de las transiciones temporales inconsistentes propias del audio urbano. Además, la CNN resulta más eficiente computacionalmente que el CRNN y más estable que el LSTM, lo que la convierte en una candidata ideal para un sistema de inferencia en producción. En Amazon SageMaker, donde el costo por llamada, la latencia y la escalabilidad son factores clave, un modelo que combina alto rendimiento y eficiencia es preferible frente a modelos más pesados o más inestables.

Finalmente, la decisión de desplegar la CNN en SageMaker no se basó únicamente en tener la mejor métrica, sino en tener el mejor equilibrio entre precisión en validación, estabilidad, costo computacional y simplicidad de integración.

Despliegue con SageMaker

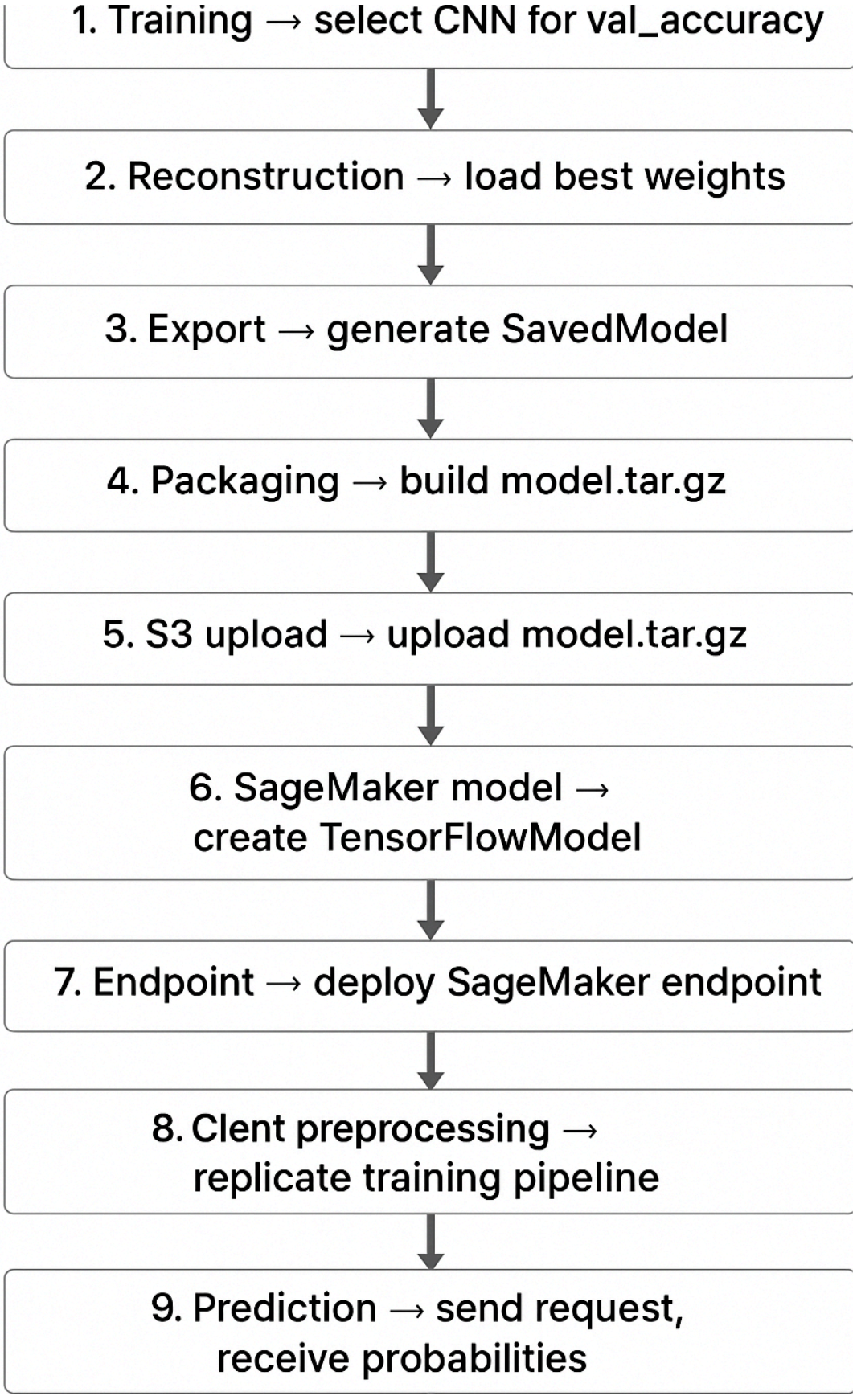
El despliegue del mejor modelo en Amazon SageMaker se construyó sobre una separación clara entre las etapas de preprocesamiento, la arquitectura entrenada y el servicio final de inferencia. Como punto de partida, se seleccionó la CNN debido a su superior rendimiento en validación, pero SageMaker no recibe un archivo de Keras directamente: requiere un artefacto empaquetado bajo el estándar SavedModel, junto con todos los metadatos necesarios para reproducir el flujo de entrada que el modelo espera. Por esa razón, el primer paso consistió en reconstruir la arquitectura desde cero y cargar únicamente los pesos optimizados, garantizando que el modelo exportado sea idéntico al que obtuvo la mejor puntuación. Esta reconstrucción evitó inconsistencias y permitió un empaquetado limpio y reproducible.

El proceso de exportación creó inicialmente un SavedModel dentro de una carpeta temporal, pero SageMaker exige una estructura muy específica: un directorio raíz con una carpeta “1/” que contenga saved_model.pb y la subcarpeta variables/. Este formato corresponde a un sistema de versionamiento interno que SageMaker utiliza para gestionar las revisiones de modelos. Además, se añadieron al nivel raíz otros elementos críticos: parámetros de normalización (mean.npy, std.npy), metadatos de despliegue (deploy_metadata.json) y el archivo preprocessing_params.json proveniente del notebook original. Esto asegura que cualquier cliente o microservicio que consuma el modelo conozca exactamente las transformaciones aplicadas durante el entrenamiento, evitando desalineaciones entre los datos de entrada reales y el espacio donde el modelo aprendió a operar. Una vez ensamblada toda la estructura, se empaquetó el directorio completo como modelo.tar.gz, el archivo que SageMaker espera para crear un endpoint.

Con el archivo comprimido dentro de un bucket de S3, SageMaker pudo instanciar un modelo TensorFlow a través de la clase TensorFlowModel. Este paso no entrena ni modifica nada: simplemente enlaza el artefacto alojado en S3 con una imagen de TensorFlow lista para servir. Para desplegarlo, se creó un endpoint con una instancia ml.c5.xlarge, suficiente para procesar

espectrogramas sin generar latencias perceptibles y con un costo computacional razonable. El sistema abrió un predictor remoto accesible desde cualquier cliente autorizado, exponiendo un servicio completamente administrado que escalará automáticamente según la carga.

Finalmente, el servicio de inferencia se integró en un notebook de SageMaker utilizando boto3. El preprocesamiento se ejecuta antes de invocar el endpoint, replicando exactamente el pipeline utilizado en el entrenamiento: carga y resampleo del audio, conversión a mono, extracción del espectrograma Mel, conversión a decibelios, normalización, padding/cropping a 160 frames y reestructuración final a (128,160,1). Esta simetría es crucial: si el modelo recibe entradas que no coinciden con las del entrenamiento, incluso un modelo robusto fallará. La predicción se envía como JSON y el endpoint devuelve un vector de probabilidades. Se aplicó un umbral de confianza del 0.6 para declarar ciertas predicciones como “indeterminadas”, priorizando confiabilidad sobre completitud.



Resultado de las predicciones en el EndPoint

Se realizaron pruebas con audios externos no pertenecientes al dataset. Estas pruebas permiten medir qué tan bien generaliza la CNN fuera del dominio controlado del conjunto de entrenamiento.

Los resultados evidenciaron una brecha significativa entre el rendimiento en test (69.41%) y el comportamiento con audios reales, especialmente cuando provienen de ambientes urbanos más complejos o formatos distintos.

En la primera prueba, un audio de tráfico vehicular fue clasificado como drilling con una probabilidad de aproximadamente 0.648. Este error tiene una explicación directa: el tráfico urbano presenta una mezcla amplia de energía en frecuencias bajas y medias, patrones repetitivos de motor y ruido ambiental estacionario. Desde la perspectiva del espectrograma, estos rasgos se parecen más a las vibraciones sostenidas y ricas en frecuencias bajas del drilling que a cualquier otra clase del dataset. Esto confirma un punto crítico: el modelo tiende a forzar las predicciones hacía clases que existían en UrbanSound8K, incluso cuando el sonido real no pertenece a ninguna de ellas. En otras palabras, su capacidad para decir “no conozco este sonido” es limitada, y depender de un umbral de 0.6 solo mitiga parcialmente el problema.

En la segunda prueba, un audio real de disparo fue clasificado como street_music con una probabilidad extremadamente alta (0.975). Esto revela otro fenómeno: si el clip no tiene la estructura impulsiva clara de los disparos del dataset —por ejemplo, si el sonido tiene reverberación, distorsión, mezcla con ruido, o es de menor calidad— el modelo deja de reconocer los rasgos que asocia con gun_shot y desplaza la predicción hacia la clase más “flexible” del dataset, que suele ser street_music. Esta clase actúa como un contenedor de patrones acústicos complejos, ricos en armónicos o con

energía dispersa en múltiples bandas. Es una consecuencia típica en modelos CNN entrenados con datos limitados: una clase muy diversa termina capturando demasiados casos ambiguos.

Finalmente, una tercera prueba con un audio de gunshot en formato MP3 fue clasificada como dog_bark con probabilidad 0.99999547, pese a que en esta ocasión el espectrograma sí contenía un pico impulsivo. El problema aquí no fue el contenido acústico, sino el formato: la compresión MP3 altera la envolvente temporal y elimina componentes de alta frecuencia, afectando exactamente las características que la CNN aprendió a reconocer como propias de gun_shot. La red neuronal no está preparada para compensar distorsiones introducidas por códecs con pérdida, por lo que interpreta el sonido comprimido como otro evento breve e impulsivo que sí existe en el dataset: dog_bark.

En conjunto, estas pruebas demuestran que el modelo desplegado funciona bien dentro del dominio de entrenamiento, pero presenta limitaciones severas cuando enfrenta sonidos reales que difieren en calidad, mezcla, reverberación o formato. Este comportamiento confirma la importancia de calibrar probabilidades, refinar el pipeline de preprocesamiento y considerar técnicas de robustez adicionales para acercar el rendimiento del endpoint al observado en validación.

Conclusión

El desarrollo del sistema de clasificación de sonidos urbanos basado en deep learning permitió recorrer de forma completa y aplicada el ciclo moderno de construcción de modelos: desde el preprocesamiento de audio y la creación de un modelo base, hasta la experimentación con distintas arquitecturas y el despliegue final en un entorno productivo mediante Amazon SageMaker. Los resultados experimentales demostraron diferencias claras en capacidad de representación entre los enfoques probados. Mientras que el modelo baseline alcanzó un rendimiento limitado (val_acc ≈ 0.25), la arquitectura CNN mostró una mejora contundente, logrando la mejor generalización entre los modelos comparados (val_acc ≈ 0.74 y test_acc ≈ 0.69). Esta brecha frente a alternativas como LSTM

o CRNN confirma que, para el dominio específico de UrbanSound8K, la información discriminante está principalmente en patrones espectrales locales más que en dependencias temporales de largo alcance.

La evaluación en test y el análisis detallado del classification report evidenciaron fortalezas importantes —como la alta precisión en clases impulsivas como gun_shot y la excelente separación de jackhammer— pero también limitaciones relevantes, especialmente en clases con mayor variabilidad acústica (engine_idling, siren) o con ruido de fondo complejo. Las pruebas finales sobre audios externos no pertenecientes al dataset permitieron contrastar el comportamiento real del sistema: el modelo tiende a cometer errores cuando el audio presenta mezcla de fuentes, formatos comprimidos o características fuera del dominio original. Estos resultados validan la decisión de incorporar un umbral de confianza (0.6) para evitar predicciones forzadas y retornar la categoría “indeterminado” en casos ambiguos.

El despliegue en SageMaker consolidó la solución en un entorno capaz de procesar audio en producción, integrando de forma modular las etapas de normalización, extracción de mel-spectrogramas y clasificación, y empaquetado adecuadamente el modelo bajo el formato requerido por TensorFlow Serving. Esta última fase confirma la viabilidad operativa del sistema y demuestra que el modelo puede ser consumido como servicio mediante un endpoint escalable.

En conjunto, el proyecto cumplió los objetivos planteados: se construyó un pipeline completo de aprendizaje profundo, se seleccionó rigurosamente el mejor modelo mediante comparación basada en métricas de validación, se analizaron los errores de forma crítica y se implementó un despliegue funcional. Aunque existen oportunidades claras de mejora —mayor robustez frente a variaciones acústicas, ampliación del dataset, técnicas avanzadas de regularización o data augmentation— el

trabajo realizado establece una base sólida tanto técnica como metodológica para continuar evolucionando el sistema hacia aplicaciones reales más exigentes.

Referencias

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/
- Amazon Web Services. (2024). Amazon SageMaker developer guide. AWS Documentation. https://docs.aws.amazon.com/sagemaker/
- Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.
- Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017). Convolutional recurrent neural networks for music classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2392–2396). IEEE. https://doi.org/10.1109/ICASSP.2017.7952585
- Chollet, F. (2015). Keras [Software]. GitHub. https://github.com/keras-team/keras
- Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.). O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/
- Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., & Platt, D. (2017). CNN architectures for large-scale audio classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 131–135). IEEE. https://doi.org/10.1109/ICASSP.2017.7952132
- McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In Proceedings of the 14th Python in Science Conference (pp. 18–25). https://librosa.org/
- Ng, A. (2019). Machine learning yearning. Deeplearning.ai.

- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Wright, R. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems (Vol. 32). https://pytorch.org/
- Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1–6). IEEE. https://doi.org/10.1109/MLSP.2015.7324337
- Rabiner, L., & Schafer, R. (2011). Theory and applications of digital speech processing. Pearson.
- Salamon, J., Jacoby, C., & Bello, J. P. (2014). A dataset and taxonomy for urban sound research. In Proceedings of the 22nd ACM International Conference on Multimedia (pp. 1041–1044). ACM. https://doi.org/10.1145/2647868.2655045
- Salamon, J., Jacoby, C., & Bello, J. P. (2014). UrbanSound8K: A dataset of urban audio recordings [Dataset]. Zenodo. https://zenodo.org/record/1203745
- Smith, J. O. (2011). Spectral audio signal processing. W3K Publishing. https://ccrma.stanford.edu/~jos/sasp/