Proyecto 1: Implementación de redes neuronales para el reconocimiento de voz a partir de espectogramas

Javier Alonso Rojas Rojas
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
javrojas@estudiantec.cr

Brandon Emmanuel Sánchez Araya
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
brandon01sanchez@estudiantec.cr

***Abstract*—Este proyecto compara tres arquitecturas convolucionales para el reconocimiento de sonidos ambientales a partir de espectrogramas de Mel del conjunto ESC-50: LeNet-5, ResNet-18 adaptada a audio y LeNet-5 con Dropout. Los audios se transformaron en espectrogramas en escala de grises (224×224) y se entrenaron en dos escenarios: datos base (*raw*) y datos aumentados mediante ruido gaussiano, *time-stretch* y *pitch-shift*. Con datos *raw*, LeNet-5 alcanzó 39.5%, ResNet-18 un 69.0% y LeNet-5+Dropout un 34.0% de precisión en prueba. Con datos aumentados, LeNet-5 subió a 86.5%, LeNet-5+Dropout a 86.0% y ResNet-18 logró el mejor resultado con 96.8% (F1≈0.97). Los resultados evidencian que la combinación de arquitecturas residuales y aumento de datos mejora significativamente la generalización y estabilidad en la clasificación de audio.**

I. Introducción

El reconocimiento automático de sonidos ambientales ha tomado importancia en los últimos años por sus aplicaciones en vigilancia, monitorización ambiental y sistemas inteligentes. En lugar de trabajar con las ondas sonoras directamente, los modelos de deep learning suelen procesar representaciones visuales como los *espectrogramas de Mel*, que describen la energía del sonido a lo largo del tiempo y la frecuencia [1].

En este proyecto se utiliza el conjunto de datos ESC-50, que contiene 2 000 clips de audio de 5 segundos distribuidos en 50 clases balanceadas de sonidos cotidianos [1]. El objetivo es comparar tres arquitecturas convolucionales: LeNet-5, como modelo base, la combinación de LeNet-5 con Dropout y una versión adaptada de ResNet-18 caracterizada por bloques residuales que facilitan el entrenamiento de redes más profundas y estables [2].

También se analiza el efecto de la técnica de aumento de datos, que introduce enmascaramientos aleatorios de tiempo y frecuencia en los espectrogramas para mejorar la generalización [3]. Con ello, se busca evaluar cómo la arquitectura y el preprocesamiento influyen en la capacidad de las redes convolucionales para reconocer patrones sonoros.

Este documento detalla el proceso completo, desde la generación de los espectrogramas y la implementación de los modelos, hasta los resultados experimentales y el análisis comparativo. A través de esta experimentación, se busca comprender de manera práctica cómo la arquitectura y el preprocesamiento influyen en la capacidad de las redes convolucionales para aprender representaciones útiles del sonido.

II. Descripción del dataset

Para el desarrollo de este proyecto se utilizó el conjunto de datos *ESC-50*, un estándar empleado para la clasificación de sonidos ambientales. Este dataset fue propuesto por Piczak en 2015 y contiene un total de 2 000 clips de audio en formato WAV, cada uno con una duración de cinco segundos y una frecuencia de muestreo de 44.1 kHz [1].

Las grabaciones se encuentran organizadas en 50 clases balanceadas, con 40 muestras por clase. Dichas clases abarcan una amplia variedad de sonidos del entorno, agrupados en cinco grandes categorías: animales, sonidos naturales, humanos no verbales, domésticos y de entorno urbano. Esta diversidad lo convierte en un conjunto perfecto para evaluar la capacidad de generalización de modelos de deep learning en audio ambiental.

Cada clip se acompaña de metadatos que incluyen el nombre del archivo, la clase correspondiente y el número de *fold* asignado para validación cruzada. En este trabajo, los *folds* 1, 2 y 3 se utilizaron para entrenamiento, el 4 para validación y el 5 para prueba, siguiendo la división sugerida por el autor del dataset.

III. Diseño de la arquitectura

El diseño de la arquitectura de red constituye un aspecto fundamental en el desempeño de los modelos de clasificación de audio basados en deep learning. En este proyecto se implementaron y compararon tres arquitecturas convolucionales con distinto nivel de complejidad, con el propósito de analizar cómo la profundidad y el tipo de bloques influyen en la extracción de características a partir de espectrogramas.

A. Modelo A – LeNet-5

El primer modelo implementado corresponde a una red convolucional clásica basada en la arquitectura *LeNet-5*, propuesta

por LeCun et al. (1998) [4]. Esta arquitectura sentó las bases del deep learning moderno al introducir un flujo jerárquico de convolución, activación no lineal y capas totalmente conectadas, originalmente diseñado para el reconocimiento de dígitos manuscritos. Su simplicidad y eficiencia la convierten en una línea base ideal para la experimentación y comparación con arquitecturas más complejas.

La elección de *LeNet-5* en este trabajo se fundamenta en su bajo costo computacional y en su capacidad de generalización en tareas con datos visuales estructurados, como los espectrogramas de audio. Además, permite observar de forma controlada el comportamiento del entrenamiento y la influencia de las funciones de activación en un entorno de red convolucional pura.

La arquitectura utilizada conserva la estructura original de *LeNet-5*, compuesta por dos bloques convolucionales seguidos de una etapa de clasificación densa. Cada bloque incluye una capa convolucional con activación *tanh* y una operación de agrupamiento máximo (*MaxPooling*) que reduce la dimensionalidad espacial de las características extraídas. En esta implementación, las capas convolucionales emplean 6 y 16 filtros respectivamente, mientras que las capas densas contienen 120, 84 y 50 neuronas, siendo esta última el número de clases del conjunto de datos.

No se utilizaron técnicas modernas de regularización como *Batch Normalization* o *Dropout*, con el objetivo de mantener la fidelidad al diseño original y establecer un punto de comparación directo con modelos más profundos y regularizados. La activación *tanh* se aplicó en todas las capas ocultas, y los pesos fueron inicializados mediante el método de Xavier, apropiado para funciones de activación simétricas.

Como se observa en la Figura **??**, el Modelo A sigue una estructura secuencial simple compuesta por dos bloques convolucionales y tres capas densas.

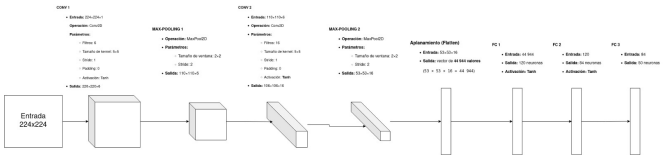


Fig. 1: Diagrama general de la arquitectura ResNet-18 adaptada para espectrogramas de audio.

B. Modelo B – ResNet-18 Audio

El segundo modelo implementado corresponde a una red convolucional residual basada en la arquitectura *ResNet-18*, propuesta originalmente por He et al. [2]. Este tipo de redes se caracteriza por el uso de bloques residuales, los cuales incorporan conexiones de salto (*skip connections*) que permiten que el gradiente fluya directamente hacia capas anteriores. Gracias a esta estructura, es posible entrenar redes más profundas sin sufrir los problemas de degradación o saturación que afectan a las CNN tradicionales.

La elección de *ResNet-18* se fundamenta en su eficacia demostrada en tareas de clasificación tanto de imágenes como de espectrogramas, donde los patrones espaciales tiempo-frecuencia presentan similitudes con las texturas visuales. Diversos estudios han mostrado que las redes residuales capturan de forma robusta las estructuras locales y globales del espectrograma, mejorando la generalización frente a variaciones en el dominio del audio [5], [6].

En esta implementación, la red fue adaptada para procesar espectrogramas en escala de grises de 224 × 224 píxeles. Se sustituyó la capa inicial de convolución por un filtro 3×3 con un solo canal de entrada y sin la etapa de max pooling temprana, con el fin de preservar mayor información espacial en las frecuencias bajas. La arquitectura mantiene la configuración clásica de ResNet-18 con cuatro etapas secuenciales que emplean 2 bloques residuales cada una, con profundidades de 64, 128, 256 y 512 canales.

Cada bloque residual está compuesto por dos convoluciones 3×3 seguidas de normalización por lotes (*Batch Normalization*) y activación ReLU. Cuando es necesario ajustar la resolución o el número de canales, se utiliza una proyección 1×1 en el atajo para garantizar la compatibilidad dimensional. Tras la última etapa, un *Global Average Pooling* reduce el mapa de activaciones a un vector de 512 características, que alimenta la capa completamente conectada de salida con 50 neuronas, correspondientes a las clases del conjunto ESC-50.

Todos los pesos fueron inicializados mediante el método de He (*Kaiming Normal*), apropiado para activaciones ReLU en redes profundas, mientras que las capas de normalización se inicializaron con pesos unitarios y sesgos nulos. Esta configuración busca equilibrar la estabilidad numérica y la capacidad de aprendizaje, favoreciendo la generalización sobre el sobreajuste.

Como se aprecia en la Figura 2, el flujo del Modelo B inicia con la convolución inicial y *BatchNorm*, seguida de las cuatro etapas residuales.

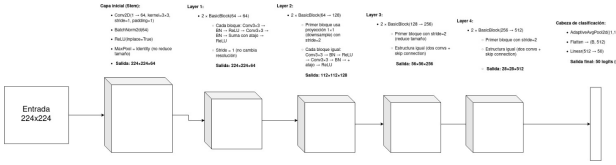


Fig. 2: Diagrama general de la arquitectura ResNet-18 adaptada para espectrogramas de audio.

C. Modelo C – LeNet-5 con Dropout

El tercer modelo propuesto corresponde a una versión regularizada del *LeNet-5* clásico, en la que se incorpora la técnica de *Dropout* con el objetivo de reducir el sobreajuste y mejorar la capacidad de generalización del modelo. Esta variante, denominada *LeNet-5 Dropout*, mantiene la misma estructura base de convolución y agrupamiento que el Modelo A, pero introduce capas de apagado estocástico en la etapa totalmente conectada.

El *Dropout* actúa desactivando aleatoriamente un porcentaje de neuronas durante el entrenamiento, evitando la coadaptación excesiva entre ellas y fomentando una representación

más robusta de los datos [7]. En esta implementación, se aplicaron tasas de desactivación del 20% (p = 0.2) antes y entre las capas densas, de forma que cada iteración entrena una subred distinta del modelo global.

La arquitectura se mantiene fiel al diseño original de *LeNet-5*, con dos bloques convolucionales que extraen características jerárquicas mediante filtros de 5×5, activaciones *tanh* y operaciones de *MaxPooling* para reducir la dimensionalidad. Posteriormente, las características se aplanan y son procesadas por tres capas densas de 120, 84 y 50 neuronas respectivamente, siendo esta última la encargada de la clasificación de las 50 clases del conjunto ESC-50.

Esta configuración logra un equilibrio entre simplicidad y regularización, manteniendo la eficiencia computacional del Modelo A mientras mitiga los efectos de sobreentrenamiento en escenarios con datos limitados.

IV. Preprocesamiento de los datos

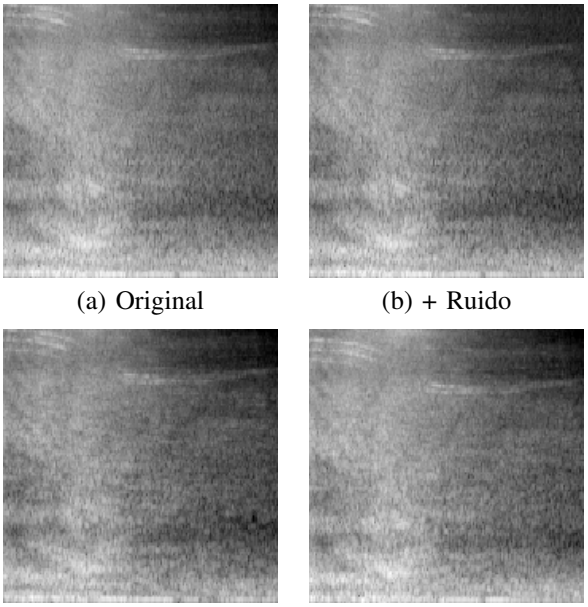
El preprocesamiento de los datos tuvo como objetivo convertir los clips de audio del conjunto ESC-50 en representaciones visuales adecuadas para redes convolucionales. Para ello, cada archivo de audio en formato WAV fue procesado con la biblioteca *librosa* para generar su correspondiente espectrograma de Mel.

Cada señal se cargó con su frecuencia de muestreo original y se transformó en un espectrograma con 128 bandas de frecuencia, limitando el rango máximo a 8 kHz. Posteriormente, el espectrograma se convirtió a escala logarítmica (decibelios) y se guardó como imagen en escala de grises con resolución de 224×224 píxeles. Las imágenes se organizaron en carpetas por clase, correspondientes a las categorías del conjunto ESC-50.

Además, se generó un conjunto aumentado aplicando tres transformaciones acústicas directamente sobre la señal de audio:

- **Ruido gaussiano:** se añadió ruido blanco con un factor de 0.005 respecto a la amplitud original.
- **Time stretch:** modificación aleatoria de la velocidad de reproducción entre 0.9 y 1.1.
- **Pitch shift:** desplazamiento tonal aleatorio entre –2 y +2 semitonos.

La Figura 3 muestra un ejemplo visual del proceso de aumento de datos aplicado sobre un clip del conjunto ESC-50, donde se observa cómo las transformaciones modifican las estructuras de tiempo y frecuencia manteniendo la coherencia acústica general del sonido.



(a) Original (b) + Ruido

(c) Time Stretch (d) Pitch Shift

Fig. 3: Ejemplo de los espectrogramas generados durante el preprocesamiento. La versión (a) corresponde al audio original, mientras que (b), (c) y (d) muestran las versiones aumentadas con ruido gaussiano, modificación temporal y desplazamiento tonal, respectivamente.

V. Entrenamiento

El proceso de entrenamiento tuvo como objetivo comparar el desempeño de las tres arquitecturas propuestas —*LeNet-5*, *LeNet-5 con Dropout* y *ResNet-18 Audio*— bajo condiciones controladas y reproducibles. Cada modelo fue entrenado utilizando las dos versiones del conjunto de datos: la versión base, compuesta por los espectrogramas originales, y la versión aumentada. Esta comparación permitió analizar el impacto del aumento de datos sobre la capacidad de generalización y la robustez frente a variaciones en el dominio acústico.

Para garantizar un análisis equitativo entre modelos, se mantuvo una configuración experimental homogénea en términos de tamaño de lote, número de épocas, esquemas de aprendizaje y criterios de evaluación. En cada arquitectura se realizaron múltiples ejecuciones (*runs*) variando los hiperparámetros principales —tasa de aprendizaje, regularización y número de épocas— con el fin de identificar las combinaciones más efectivas.

Los experimentos se realizaron con cinco configuraciones de hiperparámetros, variando la tasa de aprendizaje (`lr`), el factor de regularización (`wd`), el tamaño de lote (`batch`) y el número de épocas (`epochs`). Cada configuración se ejecutó tanto con el conjunto base (*raw*) como con el conjunto aumentado (*augmented*), registrando las métricas correspondientes en Weights & Biases (W&B).

Los conjuntos de datos se dividieron en 80% para entrenamiento, 10% para validación y 10% para prueba mediante la función `random_split`, utilizando una semilla fija para garantizar la reproducibilidad de los resultados.

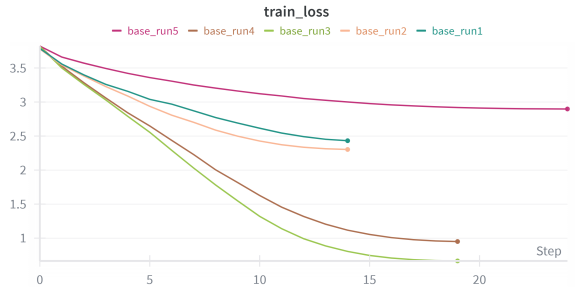
En la Tabla I se detallan los valores utilizados para cada uno de los tres modelos implementados (*LeNet-5*, *ResNet-18 Audio* y *LeNet-5 con Dropout*), tanto en los escenarios con datos base como con datos aumentados. Estas combinaciones se mantuvieron consistentes a lo largo de los experimentos, permitiendo realizar una comparación equitativa del desempeño entre arquitecturas bajo condiciones controladas.

TABLE I: Configuraciones de hiperparámetros utilizadas en los experimentos.

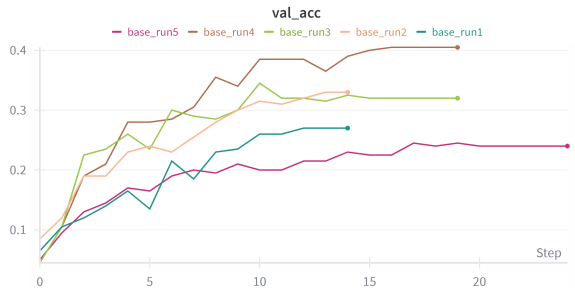
Modelo	Learning Rate (lr)	Weight Decay (wd)	Batch Size	Épocas
Modelo A – LeNet-5				
Exp. 1	1e-3	1e-4	32	15
Exp. 2	5e-4	1e-4	64	15
Exp. 3	1e-3	1e-5	64	20
Exp. 4	5e-4	1e-3	32	20
Exp. 5	1e-4	1e-4	64	25
Modelo B – ResNet-18 Audio				
Exp. 1	1e-3	1e-4	8	15
Exp. 2	5e-4	1e-4	8	15
Exp. 3	1e-3	1e-5	8	20
Exp. 4	5e-4	1e-3	8	20
Exp. 5	1e-4	1e-4	8	25
Modelo C – LeNet-5 con Dropout				
Exp. 1	1e-3	1e-4	32	15
Exp. 2	5e-4	1e-4	64	15
Exp. 3	1e-3	1e-5	64	20
Exp. 4	5e-4	1e-3	32	20
Exp. 5	1e-4	1e-4	64	25

A. *Modelo A – LeNet-5*

1) Data Raw: Se ejecutaron cinco *runs* del Modelo A (LeNet-5) sobre el conjunto base. Para esta subsección reportamos: (i) la curva de pérdida en entrenamiento (`train_loss`), (ii) la evolución de la precisión en validación (`val_acc`) y (iii) una barra comparativa de `test_acc` por *run*.



(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 4: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo A con datos *raw*.

La Figura 5 resume el rendimiento final en prueba, destacando el mejor resultado en `base_run4`.

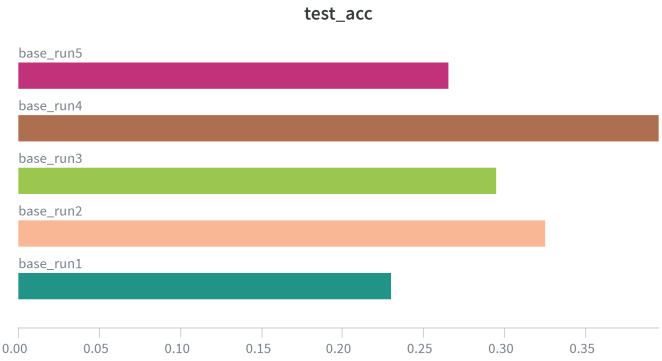


Fig. 5: Precisión en prueba (`test_acc`) por *run* del Modelo A con datos *raw*.

TABLE II: Resumen de métricas por *run* (Modelo A, datos *raw*).

Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
`base_run1`	15	0.27	0.23	0.20096	0.18489	0.26033	0.36938
`base_run2`	15	0.33	0.325	0.27498	0.27635	0.34624	0.55375
`base_run3`	20	0.32	0.295	0.30444	0.32887	0.35567	0.96062
`base_run4`	20	**0.405**	**0.395**	**0.36479**	**0.38223**	**0.42595**	**0.93688**
`base_run5`	25	0.24	0.265	0.17047	0.15259	0.27057	0.39813

2) Augmented: Se ejecutaron cinco *runs* del Modelo A (LeNet-5) utilizando el conjunto de datos aumentado mediante *time–stretch*, *pitch–shift* y adición de ruido gaussiano, tal como se describe en la Sección de Preprocesamiento. Al igual que en el escenario *raw*, considerando: (i) la pérdida de entrenamiento (`train_loss`), (ii) la precisión en validación (`val_acc`) y (iii) la precisión final en prueba (`test_acc`) por *run*.

La Figura 7 presenta la precisión final en prueba (`test_acc`) para las cinco ejecuciones, donde se observa una mejora sustancial respecto al escenario *raw*. La corrida `augmented_run4` alcanzó los mejores resultados con una precisión de validación de 0.8667 y una precisión en prueba de 0.865, mostrando una mejora absoluta de más del 45% frente al mejor resultado obtenido con los datos sin aumento.

TABLE III: Resumen de métricas por *run* (Modelo A, datos aumentados).

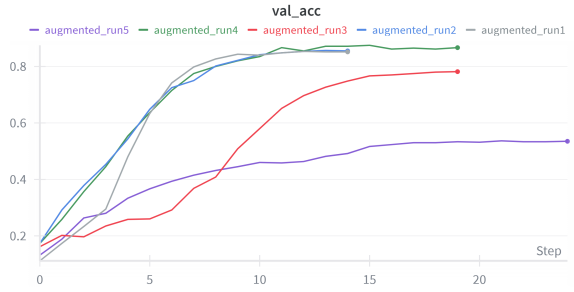
Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
`augmented_run1`	15	0.8517	0.8500	0.84596	0.85385	0.85111	0.99938
`augmented_run2`	15	0.8550	0.8600	0.84697	0.85703	0.84826	0.98604
`augmented_run3`	20	0.7817	0.7750	0.77049	0.77906	0.77961	0.95833
`augmented_run4`	20	**0.8667**	**0.8650**	**0.85926**	**0.86953**	**0.86189**	**0.99958**
`augmented_run5`	25	0.5350	0.5683	0.49803	0.54981	0.53593	0.70104

B. *Modelo B – ResNet-18 Audio*

1) Data Raw: Se ejecutaron cinco *runs* del Modelo B sobre el conjunto base (sin aumentos). Para esta subsección reportamos: (i) la curva de pérdida en entrenamiento (`train_loss`), (ii) la evolución de la precisión en validación (`val_acc`) y (iii) una barra comparativa de `test_acc` por *run*.

La Figura 9 resume el rendimiento final en prueba, destacando el mejor resultado en `base_run5`.

(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 6: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo A con datos aumentados.

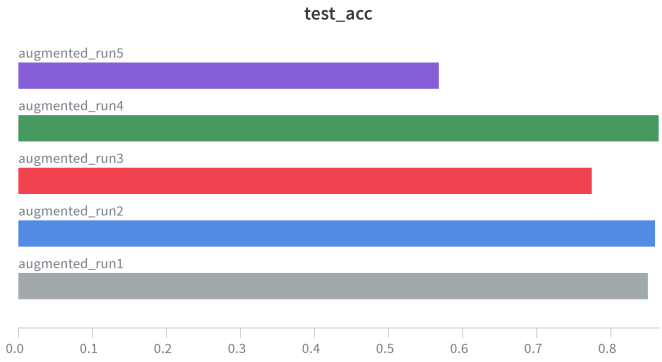
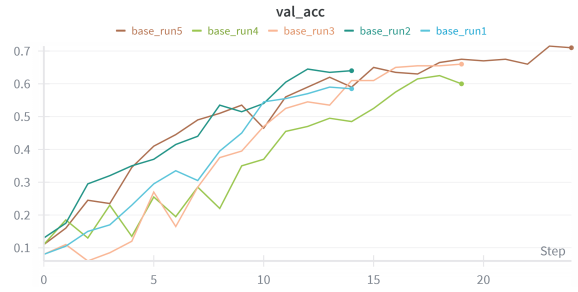


Fig. 7: Precisión en prueba (`test_acc`) por *run* del Modelo A con datos aumentados.

TABLE IV: Resumen de métricas por *run* (Modelo B, datos *raw*).

Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
base_run1	15	0.585	0.585	0.55651	0.61307	0.59869	0.60500
base_run2	15	0.640	0.620	0.60598	0.64486	0.65136	0.68563
base_run3	20	0.660	0.670	0.63084	0.66657	0.67498	0.69500
base_run4	20	0.600	0.575	0.55781	0.61557	0.60114	0.65750
base_run5	25	**0.710**	**0.690**	**0.66552**	**0.70098**	**0.69536**	**0.79812**

(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 8: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo B con datos *raw*.

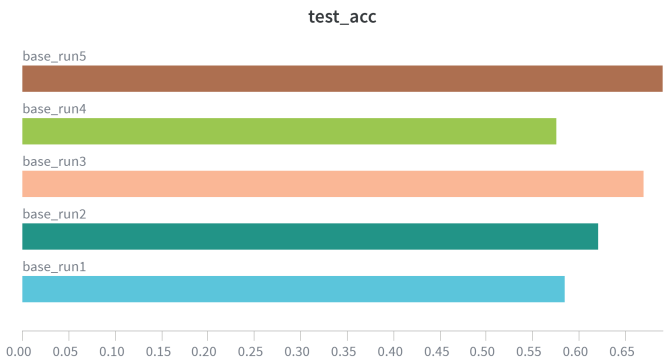


Fig. 9: Precisión en prueba (`test_acc`) por *run* del Modelo B con datos *raw*.

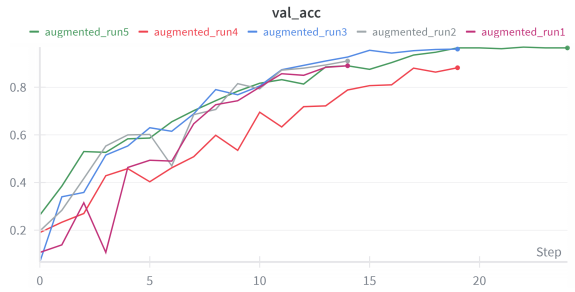
2) Augmented: Se ejecutaron cinco *runs* del Modelo B sobre el conjunto aumentado, entrenado con los espectrogramas generados mediante las técnicas de aumento descritas previamente. Para esta subsección se presentan: (i) la curva de pérdida en entrenamiento (`train_loss`), (ii) la evolución de la precisión en validación (`val_acc`) y (iii) una barra comparativa de `test_acc` por *run*.

La Figura 11 resume el rendimiento final en el conjunto de prueba, destacando el mejor resultado en `augmented_run5`.

C. Modelo C – LeNet-5 con Dropout

1) Data Raw: Se ejecutaron cinco *runs* del Modelo C sobre el conjunto base (sin aumentos). Para esta subsección reporta-

(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 10: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo B con datos *augmented*.

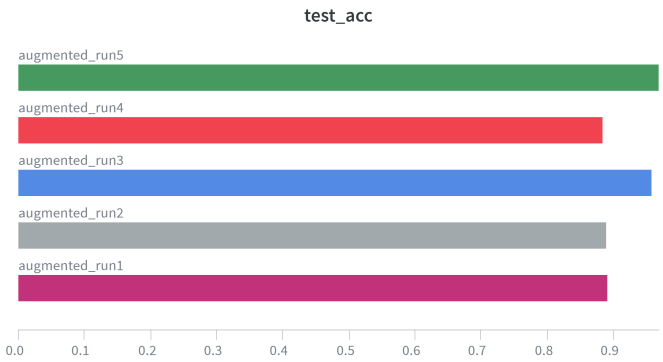
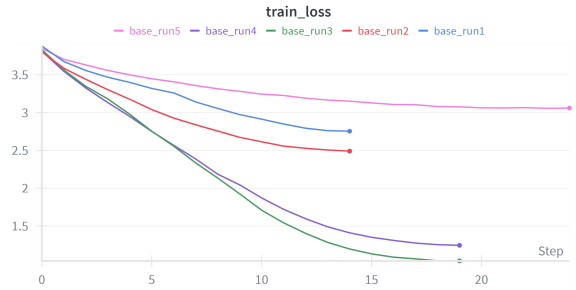


Fig. 11: Precisión en prueba (`test_acc`) por *run* del Modelo B con datos *augmented*.

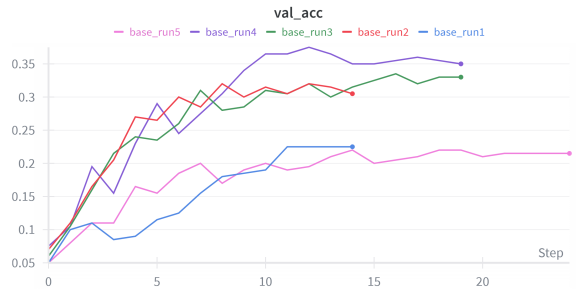
TABLE V: Resumen de métricas por *run* (Modelo B, datos *augmented*).

Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
augmented_run1	15	0.890	0.890	0.87802	0.89087	0.88431	0.90771
augmented_run2	15	0.910	0.888	0.88056	0.88786	0.88610	0.90833
augmented_run3	20	0.960	0.958	0.95452	0.95743	0.95472	0.97875
augmented_run4	20	0.882	0.883	0.87375	0.88913	0.87908	0.89354
augmented_run5	25	**0.965**	**0.968**	**0.96582**	**0.96690**	**0.96878**	**0.87583**

mos: (i) la curva de pérdida en entrenamiento (`train_loss`), (ii) la evolución de la precisión en validación (`val_acc`) y (iii) una barra comparativa de `test_acc` por *run*.



(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 12: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo C con datos *raw*.

La Figura 13 resume el rendimiento final en prueba, destacando el mejor resultado en `base_run4`.

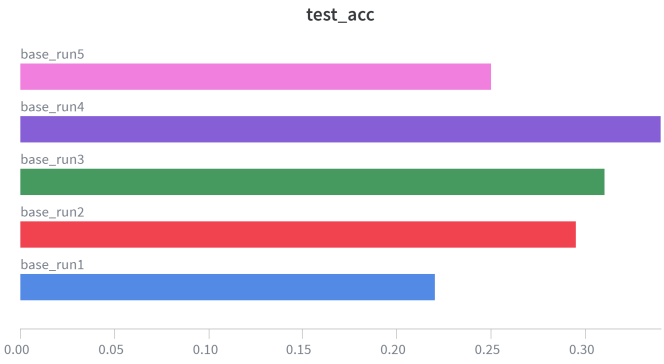


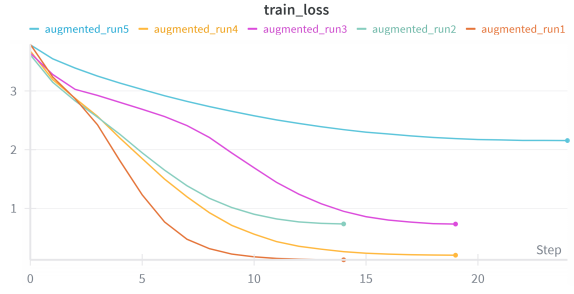
Fig. 13: Precisión en prueba (`test_acc`) por *run* del Modelo C con datos *raw*.

2) Augmented: Se ejecutaron cinco *runs* del Modelo C con datos aumentados. Para esta subsección reportamos: (i) la curva de pérdida en entrenamiento (`train_loss`), (ii) la evolución de la precisión en validación (`val_acc`) y (iii) una barra comparativa de `test_acc` por *run*.

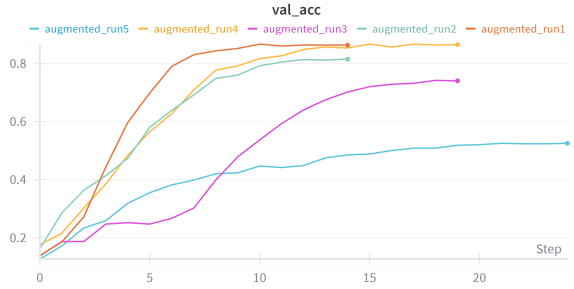
La Figura 15 resume el rendimiento final en prueba, destacando el mejor resultado en `augmented_run4`.

TABLE VI: Resumen de métricas por *run* (Modelo C, datos *raw*).

Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
base_run1	15	0.225	0.220	0.15453	0.13899	0.23543	0.27813
base_run2	15	0.305	0.295	0.23197	0.21065	0.30190	0.51313
base_run3	20	0.330	0.310	0.30094	0.30968	0.37238	0.91687
base_run4	20	**0.350**	**0.340**	**0.32191**	**0.33442**	**0.38652**	**0.88750**
base_run5	25	0.215	0.250	0.16033	0.16169	0.25086	0.33313



(a) `train_loss` por época.



(b) `val_acc` por época.

Fig. 14: Evolución de la pérdida y precisión de validación durante el entrenamiento del Modelo C con datos *augmented*.

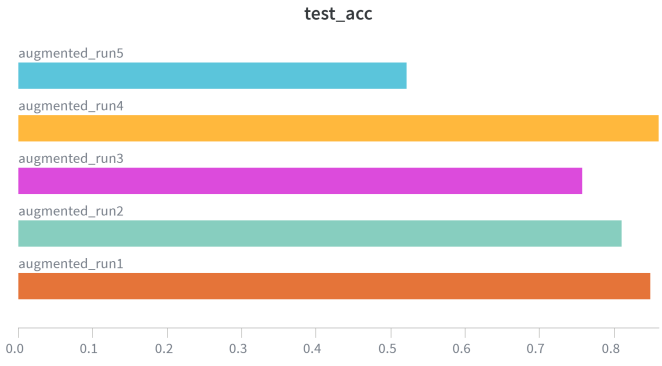


Fig. 15: Precisión en prueba (`test_acc`) por *run* del Modelo C con datos *augmented*.

TABLE VII: Resumen de métricas por *run* (Modelo C, datos *augmented*).

Run	Épocas	Val Acc	Test Acc	F1	Precisión	Recall	Train Acc
augmented_run1	15	0.863	0.848	0.85466	0.86356	0.86035	0.99896
augmented_run2	15	0.815	0.810	0.80258	0.81039	0.80942	0.94333
augmented_run3	20	0.740	0.757	0.72707	0.73849	0.74076	0.92333
augmented_run4	20	**0.865**	**0.860**	**0.85887**	**0.86932**	**0.86129**	**0.99917**
augmented_run5	25	0.525	0.522	0.48446	0.54446	0.52566	0.63500

VI. Evaluación de modelos

El análisis comparativo de los tres modelos desarrollados permite valorar el impacto de la arquitectura y del aumento de datos sobre la capacidad de clasificación de los espectrogramas de audio. En esta sección se discuten los resultados obtenidos en los diferentes escenarios, considerando tanto las métricas cuantitativas (precisión, F1, *recall*, *precision*) como los patrones de convergencia observados durante el entrenamiento.

A. Desempeño general por arquitectura

Los resultados muestran diferencias claras entre los modelos evaluados. El **Modelo A (LeNet-5)** actuó como línea base y logró un desempeño moderado con los datos sin aumento, alcanzando una *test accuracy* máxima de 39.5%. Sin embargo, al aplicar técnicas de aumento de datos (ruido gaussiano, *time stretch* y *pitch shift*), su rendimiento se elevó de manera significativa, con valores de *test accuracy* cercanos al 86.5% y un F1 de 0.86. Esto evidencia que la arquitectura clásica, aunque limitada en profundidad, puede generalizar adecuadamente si se le provee un conjunto de datos más diverso y robusto.

El **Modelo B (ResNet-18 Audio)** presentó un rendimiento consistentemente superior en ambos escenarios, destacando por su estabilidad y rápida convergencia. Con los datos base alcanzó hasta 69.0% de precisión en prueba, mientras que con datos aumentados superó el 96.8%, junto con un F1 de 0.97. Estos resultados confirman la eficacia de las conexiones residuales para preservar el flujo del gradiente y extraer representaciones más profundas del espectrograma. Además, la menor diferencia entre las métricas de entrenamiento y validación sugiere una buena capacidad de generalización.

El Modelo C (LeNet-5 con Dropout) no incrementó de forma sustancial la precisión respecto al Modelo A, pero sí mejoró la estabilidad del entrenamiento y la robustez frente al sobreajuste. Cuando ambos se entrenaron con datos aumentados, el desempeño fue prácticamente idéntico, lo que confirma que la regularización por Dropout puede compensar parte del beneficio del aumento de datos, aportando una mayor consistencia en las métricas.

B. Efecto del aumento de datos

El aumento de datos tuvo un efecto positivo en todos los modelos, incrementando la precisión promedio de prueba entre un 40% y un 60% respecto a sus versiones sin aumento. En los tres casos, el entrenamiento con datos aumentados generó curvas de pérdida más estables y una convergencia más rápida, reduciendo las fluctuaciones en `val_loss`. Este comportamiento sugiere que las transformaciones acústicas introducidas (ruido, estiramiento temporal y desplazamiento

tonal) aportaron suficiente variabilidad para mejorar la robustez del modelo ante condiciones sonoras distintas a las del conjunto original.

C. Análisis de estabilidad y convergencia

Las curvas de entrenamiento reflejan comportamientos característicos de cada arquitectura. LeNet-5 mostró una disminución progresiva pero más lenta del `train_loss`, mientras que ResNet-18 alcanzó la estabilidad alrededor de la época 15, indicando una convergencia eficiente y estable. El uso de Batch Normalization y activaciones ReLU contribuyó a la fluidez del gradiente, evitando oscilaciones bruscas. En el caso del LeNet-5 con Dropout, la convergencia fue más irregular en las primeras épocas, pero logró estabilizarse posteriormente sin sobreajuste marcado, lo cual valida el efecto regularizador del método.

D. Comparación de métricas

El comportamiento de las métricas *precision*, *recall* y F1 muestra que los modelos aumentados lograron un equilibrio mayor entre sensibilidad y especificidad. En los escenarios sin aumento, los valores de *recall* eran significativamente inferiores a los de *precision*, lo que indica que los modelos eran conservadores y omitían algunas clases. Con el aumento de datos, ambas métricas se aproximaron, reflejando una mejor calibración de las predicciones y una mayor capacidad para detectar correctamente ejemplos positivos.

En términos globales, el **Modelo B con datos aumentados** fue el que presentó el mejor desempeño, alcanzando los valores más altos en todas las métricas evaluadas, junto con una convergencia rápida y estable. El **Modelo A** evidenció la mayor mejora relativa tras el aumento de datos, pasando de un rendimiento bajo a uno competitivo, mientras que el **Modelo C** se destacó como una alternativa balanceada entre simplicidad y capacidad de generalización.

En conjunto, los resultados demuestran que tanto la arquitectura como la estrategia de preprocesamiento influyen de forma determinante en la eficacia del reconocimiento de sonidos ambientales. Las técnicas de aumento de datos y la incorporación de bloques residuales se consolidan como los factores más relevantes para lograr una clasificación robusta y generalizable en este tipo de tareas.

VII. Conclusiones

El desarrollo de este proyecto permitió analizar de manera integral el comportamiento de diferentes arquitecturas de redes neuronales convolucionales aplicadas al reconocimiento de sonidos ambientales, utilizando el conjunto de datos ESC-50 como base experimental. A lo largo de las fases de preprocesamiento, entrenamiento y evaluación, se evidenció la relevancia tanto de la estructura del modelo como de la calidad y diversidad del conjunto de datos en la obtención de resultados robustos.

En primer lugar, se comprobó que la arquitectura **LeNet-5**, aunque clásica y de menor complejidad, puede alcanzar un desempeño competitivo cuando se complementa con técnicas de aumento de datos. Su precisión pasó de 39.5% con datos sin procesar a 86.5% con datos aumentados, demostrando que el enriquecimiento del conjunto de entrenamiento tiene un impacto directo en la capacidad de generalización del modelo.

En segundo lugar, la arquitectura **ResNet-18 Audio** se consolidó como la de mejor rendimiento global, alcanzando hasta un 96.8% de precisión en prueba. Esto confirma que las arquitecturas profundas con mecanismos de atajo resultan más adecuadas para tareas de clasificación compleja en el dominio espectral del audio.

Por otra parte, el **LeNet-5 con Dropout** logró un equilibrio adecuado entre simplicidad computacional y desempeño, reduciendo la sobreajuste presente en el modelo base y obteniendo mejoras notables en estabilidad. Este resultado resalta la importancia de incorporar técnicas de regularización, especialmente cuando se dispone de modelos con menor capacidad representacional.

De forma transversal, el uso de aumentos de datos demostró ser una estrategia clave para mejorar la robustez del sistema frente a variaciones acústicas. Su aplicación permitió estabilizar las curvas de pérdida, mejorar la precisión en validación y reducir la brecha entre las métricas de entrenamiento y prueba.

Los resultados obtenidos evidencian que el desempeño de los modelos de clasificación de audio depende en gran medida del equilibrio entre complejidad arquitectónica, estrategias de regularización y diversidad de los datos. Entre los enfoques analizados, la combinación de una arquitectura residual con datos aumentados ofreció la mejor relación entre rendimiento, estabilidad y capacidad de generalización, estableciendo una referencia sólida para futuras investigaciones en el área de reconocimiento de patrones acústicos.

References

[1] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 2015, pp. 1015–1018.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[3] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019, pp. 2613–2617.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] Q. Kong and Y. e. a. Cao, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, 2020, pp. 2880–2894.

[6] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *ISMIR*, 2017, pp. 141–149.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.