Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Computación

Transferencia de Aprendizaje Hacia el Modelo ResNet-50 para la Clasificación de Especies de Aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris

Transfer Learning Towards the ResNet-50 Model for the Classification of Blackish Churrin, Plain-tailed Wren and Grey-breasted Wood Wren Bird Species

Trabajo de Integración Curricular previa a la obtención del título de Ingeniero en Ciencias de la Computación

AUTOR:

Jimmy Alexander Cajamarca Escaleras

DIRECTOR:

Ing. Valeria del Rosario Herrera Salazar Mg. Sc

Loja – Ecuador

2025

Certificación de director

Loja, 03 de febrero de 2025

Ing. Valeria del Rosario Herrera Salazar, Mg. Sc.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Transferencia de Aprendizaje Hacia el Modelo ResNet-50 para la Clasificación de Especies de Aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris,** previo a la obtención del título de **Ingeniero en Ciencias de la Computación,** de autoría del estudiante: **Jimmy Alexander Cajamarca Escaleras,** con cédula de identidad **1106070889,** una vez que se determina que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
VALERIA DEL ROSARIO HERRERA SALAZAR

Ing. Valeria del Rosario Herrera Salazar, Mg. Sc.
Director del Trabajo de Integración Curricular

Autoría

Yo, **Jimmy Alexander Cajamarca Escaleras**, declaro ser autor del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de identidad: 1106070889

Fecha: 03 de febrero de 2025

Correo electrónico: jimmy.cajamarca@unl.edu.ec

Teléfono: 0991739340

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular

Yo, **Jimmy Alexander Cajamarca Escaleras**, declaro ser el autor del Trabajo de Integración Curricular denominado: **Transferencia de Aprendizaje Hacia el Modelo ResNet-50 para la Clasificación de Especies de Aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris,** autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular o de Titulación que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los tres días del mes de febrero de dos mil veinticinco.

Firma: Firmado electrónicamente por: JIMMY ALEXANDER CAJAMARCA ESCALERAS

Autor: Jimmy Alexander Cajamarca Escaleras
Cédula de identidad: 1106070889
Dirección: Loja - Ecuador
Correo electrónico: jimmy.cajamarca@unl.edu.ec
Teléfono: 0991739340

DATOS COMPLEMENTARIOS:
Director del Trabajo de Integración Curricular: Ing. Valeria del Rosario Herrera Salazar Mg. Sc.

Dedicatoria

Este trabajo de integración curricular está dedicado con profundo respeto y admiración a mis padres, Julio y Martha. Su apoyo incondicional, paciencia y enseñanzas me han permitido superar los desafíos y alcanzar este importante logro. Ellos han sido la base de mi formación y sin su amor y esfuerzo constante, este camino no habría sido posible.

A mis hermanos, Anthony y Dylan, quienes siempre me han acompañado y sido una fuente constante de motivación y energía. Como hermano mayor, confío en que este logro les sirva de inspiración, y que puedan seguir persiguiendo sus sueños con la misma dedicación y empeño.

Jimmy Alexander Cajamarca Escaleras

Agradecimiento

Primero, quiero agradecer a Dios, por darme la fortaleza, la claridad y la oportunidad de llegar hasta este importante logro. Su guía ha sido mi fuente de inspiración en cada paso de este recorrido, especialmente en los momentos de incertidumbre.

A mis padres y hermanos, por su amor incondicional, su apoyo constante y por ser mi mayor fuente de motivación. Cada sacrificio que han hecho por mí, cada palabra de aliento y cada gesto de cariño me han permitido alcanzar este sueño. Este logro es tan suyo como mío, pues sin su acompañamiento y confianza, nada de esto habría sido posible.

Mi sincero agradecimiento a mi directora, Ing. Valeria del Rosario Herrera Salazar, Mg. Sc., por su dedicación, paciencia y sabiduría. Gracias por guiarme a lo largo de este proceso, por revisar incansablemente cada detalle de este trabajo y por brindarme siempre sus valiosos consejos. Su apoyo ha sido esencial para que este proyecto fuera llevado a cabo de la mejor manera.

Agradezco también al Ing. Oscar Miguel Cumbicus Pineda, Mg. Sc., por su valiosa orientación en el campo de la inteligencia artificial. Su experiencia y conocimientos fueron clave para la parte técnica de este trabajo, y gracias a su guía, pude afrontar los desafíos con mayor claridad y confianza.

Finalmente, quiero agradecer a mis familiares, amigos y compañeros, quienes han estado presentes en cada etapa de este proceso, brindándome su apoyo y ánimo. Este trabajo no solo refleja mi esfuerzo, sino también el amor y la confianza que ellos me han otorgado.

Jimmy Alexander Cajamarca Escaleras

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Índice de Ecuaciones

Índice de Anexos

1. Título

Transferencia de aprendizaje hacia el modelo ResNet-50 para la clasificación de especies de aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

Transfer Learning Towards the ResNet-50 Model for the Classification of Blackish Churrin, Plain-tailed Wren and Grey-breasted Wood Wren Bird Species.

2. Resumen

El reconocimiento automático de vocalizaciones de aves mediante aprendizaje profundo es una herramienta importante para el monitoreo y la conservación de especies, especialmente en ecosistemas frágiles donde los métodos tradicionales, como el reconocimiento manual por expertos, son costosos y poco escalables. Este trabajo tuvo como objetivo determinar el porcentaje global de acierto al aplicar transferencia de aprendizaje al modelo ResNet-50 para clasificar las vocalizaciones de tres especies de aves ecuatorianas: Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris. Siguiendo la metodología CRISP-ML(Q), se desarrollaron tres fases principales para el primer objetivo. En la primera fase, se realizó una entrevista con el experto en Machine Learning, Ingeniero Oscar Cumbicus como contexto inicial para guiar el desarrollo del proyecto y definir su viabilidad técnica. En la segunda fase, se recopilaron 1164 grabaciones desde la plataforma Xeno-canto, las cuales fueron preprocesadas, normalizadas y transformadas en espectrogramas. Además, se aplicaron técnicas de aumento de datos para mejorar la representatividad. En la tercera fase, se ajustó el modelo ResNet-50 mediante transferencia de aprendizaje, optimizando hiperparámetros e implementando callbacks para mejorar el rendimiento y prevenir el sobreajuste. Para el segundo objetivo, se ejecutó la fase de evaluación de modelos de ML, en la que el modelo alcanzó una precisión del 91% en el conjunto de entrenamiento y del 89.33% en el conjunto de prueba. Su desempeño fue validado con audios externos mediante una interfaz web que permitió su uso por usuarios no especializados, logrando un porcentaje global de acierto del 91.19%. Estos resultados no solo demuestran la solidez del enfoque propuesto, sino que también establecen un referente en la clasificación bioacústica, ofreciendo nuevas perspectivas para el desarrollo de herramientas avanzadas en el estudio y conservación de la biodiversidad.

Palabras clave: Transferencia de aprendizaje, ResNet-50, Bioacústica, Clasificación, Aprendizaje automático, Conservación de aves.

Abstract

The automatic recognition of bird vocalizations through deep learning is an important tool for monitoring and conservation of species, especially in fragile ecosystems where traditional methods, such as manual identification by experts, are expensive and not scalable. The objective of this work was to determinate the overall percentage of accuracy when applying transfer learning to the ResNet-50 model to classify the vocalizations of three Ecuadorian bird species: Blackish Churrin, Plain-tailed Wren, and Grey-breasted Wood Wren. Following the CRISP-ML(Q) methodology, three main phases were developed for the first objective. In the first phase, an interview with Machine Learning expert, Engineer Oscar Cumbicus was conducted as an initial context to guide the development of the project and define its technical feasibility. In the second phase, 1,164 recordings were collected from the Xeno-canto platform, which were preprocessed, normalized, and transformed into spectrograms. In addition, data augmentation techniques were applied to improve representativeness. In the third phase, the ResNet-50 model was tuned by transfer learning, optimizing hyperparameters and implementing callbacks to improve performance and prevent overfitting. For the second objective, the ML model evaluation phase was executed, in which the model achieved an accuracy of 91% in the training set and 89.33% in the test set. Its performance was validated with external audios through a web interface that allowed its use by non-specialized users, achieving an overall hit rate of 91.19%. These results not only demonstrate the robustness of the proposed approach, but also establish a benchmark in bioacustic classification, offering new perspectives for the development of advanced tools in the study and conservation of biodiversity.

Keywords: Transfer learning, ResNet-50, Bioacustic, Classification, Machine learning, Bird conservation.

3. Introducción

El reconocimiento automático de vocalizaciones de aves es un campo emergente dentro de la bioacústica, que busca resolver las limitaciones asociadas con los métodos tradicionales de identificación realizados manualmente por expertos en ornitología. Estos métodos, aunque efectivos, presentan altos costos en términos de equipos especializados, personal, transporte y tiempo para la grabación y adquisición de audios en los ecosistemas naturales [1]. Además, la complejidad inherente de las vocalizaciones de aves, caracterizadas por patrones acústicos distintivos y un alto grado de variabilidad, representa un desafío significativo para los sistemas de clasificación tradicionales.

Este proyecto aborda estas limitaciones mediante la implementación de técnicas de aprendizaje profundo, con énfasis en la transferencia de aprendizaje utilizando la arquitectura ResNet-50. Este modelo, conocido por su capacidad para procesar datos visuales complejos, se adapta a la clasificación de vocalizaciones transformándolas en representaciones visuales mediante espectrogramas, aprovechando su capacidad para extraer características relevantes [2]. En este contexto, el objetivo general es determinar el porcentaje global de acierto al aplicar transferencia de aprendizaje al modelo ResNet-50 para clasificar las vocalizaciones de tres especies de aves: Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

La importancia de esta investigación radica en su aplicación directa a la conservación de especies endémicas y migratorias presentes en la región andina de Ecuador, particularmente en el Parque Nacional Podocarpus, un área de alta biodiversidad. La identificación precisa de estas especies a través de sus vocalizaciones no solo mejora la comprensión de su comportamiento y distribución, sino que también facilita el monitoreo de su estado de conservación [3]. Sin embargo, la diversidad acústica y la presencia de ruido en los entornos naturales dificultan el uso directo de modelos preentrenados en datos genéricos como ImageNet, requiriendo adaptaciones específicas mediante transferencia de aprendizaje [4].

De acuerdo con la entrevista realizada al experto Ingeniero Oscar Cumbicus (ver **Anexo *1***), los principales desafíos en la clasificación automática de vocalizaciones incluyen la presencia de ruido ambiental, la mezcla de vocalizaciones de diferentes especies y la falta de modelos especializados que capturen las características acústicas únicas de estas señales. Además, se destaca la relevancia de integrar soluciones tecnológicas que no solo optimicen el proceso de identificación, sino que también reduzcan los costos asociados a los métodos manuales.

Para estructurar este desarrollo, se adoptó la metodología CRISP-ML(Q), que permitió abordar sistemáticamente desde la recolección y preprocesamiento de datos hasta el ajuste de hiperparámetros y la evaluación de resultados. Esto incluyó la generación de espectrogramas, la normalización de amplitud y el aumento de datos para mitigar problemas como el desbalance de clases y la escasez de datos etiquetados. Además, se implementó una interfaz web para facilitar el acceso al sistema, promoviendo aplicaciones prácticas en el monitoreo y conservación de especies.

Con esta investigación, se busca proporcionar una herramienta innovadora que contribuya al monitoreo automatizado de la biodiversidad en Ecuador. Este enfoque tiene el potencial de ser replicado en otros contextos geográficos y ecológicos, fortaleciendo así las estrategias de conservación en regiones de alta diversidad biológica.

4. Marco teórico

4.1. Antecedentes

Tradicionalmente, la identificación de especies de aves ha dependido de observaciones visuales y auditivas realizadas por expertos en el campo [5]. Sin embargo, este método presenta limitaciones considerables en términos de cobertura geográfica, consistencia y la posibilidad de realizar un monitoreo continuo, especialmente en áreas remotas o de difícil acceso. Es en este contexto donde las técnicas de aprendizaje profundo, en particular la aplicación de modelos de redes neuronales convolucionales (CNN) como ResNet-50, ofrecen una alternativa prometedora [6].

La identificación y clasificación automática de especies de aves mediante el análisis de sus vocalizaciones se ha convertido en un campo de investigación emergente, con implicaciones significativas tanto para el monitoreo ecológico como para la conservación de la biodiversidad [7]. El uso de redes neuronales profundas, como las CNN, permite automatizar el monitoreo en tiempo real, lo que no solo aumenta la cobertura geográfica, sino que también facilita la recolección de datos consistentes y escalables en ecosistemas complejos. La elección de ResNet-50 como base para este estudio se fundamenta en su arquitectura profunda, compuesta por 50 capas y su capacidad demostrada para extraer características complejas en tareas de clasificación de imágenes [8]. Aunque originalmente diseñada para el procesamiento visual, la adaptabilidad de ResNet-50 a través de técnicas de transferencia de aprendizaje permite aprovechar el conocimiento adquirido sobre patrones y estructuras visuales en un dominio (clasificación de imágenes) a un dominio relacionado pero distinto (clasificación de espectrogramas de audio) [9]. Esto abre la puerta a un nuevo enfoque de clasificación bioacústica que mejora la precisión y eficiencia de los métodos anteriores utilizados en este campo.

El presente estudio se centra en la clasificación de tres especies de aves neotropicales [10]: el Churrín Negruzco (Scytalopus latrans), el Soterrey Montés de Pecho Gris (Henicorhina leucoptera) y el Soterrey Cola Pálida (Pheugopedius euophrys), todas ellas endémicas de la región andina y frecuentemente avistadas en Loja, Ecuador [11]. La selección de estas especies no es arbitraria; su presencia y abundancia actúan como indicadores clave de la salud de los ecosistemas montañosos andinos. El desarrollo de métodos automáticos y precisos para su identificación es esencial no solo para el seguimiento de estas especies, sino también para la implementación de estrategias de conservación y la gestión sostenible de estos ecosistemas biodiversos.

4.1.1. Churrín Negruzco (Scytalopus latrans)

El Churrín Negruzco es una pequeña ave insectívora que habita en los bosques montanos del oeste de América del Sur, principalmente en la región de los Andes [12]. Es conocido por su plumaje oscuro y su canto distintivo, el cual es un elemento primordial para su identificación. Su canto es repetitivo y de alta frecuencia, lo que lo convierte en un desafío acústico interesante para los modelos de clasificación. Esta especie es de hábitos terrestres y se encuentra en áreas densamente cubiertas de vegetación, lo que dificulta su observación visual, haciendo el análisis acústico un método efectivo para su estudio y monitoreo [13]. En la **Figura 1** se observa a la especie de ave Churrín Negruzco (*Scytalopus latrans)*, tomada por [David McDonald], bajo licencia CC BY-NC, disponible en Macaulay Library [14].

Figura 1: Churrín Negruzco (Scytalopus latrans)

4.1.2. Soterrey Montés de Pecho Gris (Henicorhina leucoptera)

El Soterrey Montés de Pecho Gris, es un pequeño pájaro que habita en bosques húmedos y montanos desde México hasta el norte de Perú, incluyendo las montañas de Ecuador [12]. Su canto es complejo y melodioso, caracterizado por secuencias variadas de notas, lo que lo convierte en un excelente sujeto para el estudio bioacústico. Este soterrey es territorial y su vocalización es clave para la defensa de su territorio y la comunicación con otros individuos. Su preferencia por los bosques densos y húmedos hace que su detección auditiva sea mucho más efectiva que la visual [15]. En la **Figura 2** se observa a la especie de ave Soterrey Montés de Pecho Gris (*Henicorhina leucoptera)*, tomada por [Stephan Lorenz], bajo licencia CC BY-NC, disponible en Macaulay Library [16].

Figura 2: Soterrey Montés de Pecho Gris (Henicorhina leucoptera)

4.1.3. Soterrey Cola Pálida (Pheugopedius euophrys)

El Soterrey Cola Pálida, es un ave andina que habita en bosques montanos entre los 2,500 y 3,500 metros de altitud, principalmente en Ecuador y Perú [12]. Es conocido por sus vocalizaciones complejas, especialmente los duetos entre machos y hembras, que juegan un papel importante en el comportamiento territorial y de apareamiento. Su canto claro y repetitivo, junto con su preferencia por áreas de vegetación densa, lo convierte en una especie clave para estudios de bioacústica y monitoreo a través de grabaciones [17]. En la **Figura 3** se observa a la especie de ave Soterrey Cola Pálida (*Pheugopedius euophrys)*, tomada por [Andrés Vasquez Noboa], bajo licencia CC BY-NC, disponible en Macaulay Library [18].

Figura 3: Soterrey Cola Pálida (Pheugopedius euophrys)

4.1.4. Ciencia Ciudadana en Ornitología

La Ciencia Ciudadana en Ornitología se refiere a la colaboración entre investigadores profesionales y el público general en la recopilación y análisis de datos relacionados con aves [19]. Esta participación ha permitido un acceso masivo a observaciones de campo, grabaciones de cantos, y otros tipos de datos que de otro modo serían difíciles de obtener debido a la extensión geográfica y temporal requerida. Plataformas como Xeno-canto [20], eBird [21] y otras comunidades en línea han transformado la ornitología moderna al permitir que ciudadanos de todo el mundo suban grabaciones de cantos de aves, compartan observaciones visuales, y contribuyan a bases de datos globales.

4.1.4.1. Xeno-Canto.

Xeno-canto es una plataforma en línea dedicada a compartir fotos, grabaciones de sonidos de aves de todo el mundo, creada en 2005 [22]. Está impulsada por una comunidad de científicos, ornitólogos aficionados y ciudadanos que suben grabaciones de cantos y llamadas de aves desde diversas localizaciones [23]. Este sitio web se ha convertido en uno de los recursos más importantes para la investigación en ornitología y bioacústica [24], al permitir el acceso libre y gratuito a un extenso repositorio de sonidos.

4.2. Fundamentación Teórica

4.2.1. Inteligencia Artificial

La Inteligencia Artificial (IA) es una rama de la informática que busca desarrollar sistemas capaces de realizar tareas que tradicionalmente requieren inteligencia humana, como el reconocimiento de patrones, la toma de decisiones, la percepción visual y el procesamiento del lenguaje [25]. Estos sistemas utilizan algoritmos avanzados, modelos matemáticos y técnicas de aprendizaje automático [26] para analizar grandes volúmenes de datos, aprender de ellos y adaptarse a nuevas situaciones sin necesidad de intervención humana. IA abarca desde soluciones simples basadas en reglas hasta modelos más complejos como las redes neuronales profundas, que imitan el funcionamiento del cerebro para resolver problemas complejos de manera autónoma.

4.2.2. Machine Learning

El Machine Learning (ML) o aprendizaje automático hace referencia al subcampo de la inteligencia artificial cuyo objetivo es desarrollar técnicas mediante las cuales los sistemas informáticos puedan aprender de manera autónoma a partir del reconocimiento de patrones complejos en grandes volúmenes de datos [27]. Por medio de algoritmos y modelos estadísticos, las computadoras mejoran progresivamente su desempeño en tareas específicas como clasificación, predicción y toma de decisiones [28], sin necesidad de estar explícita o rigurosamente programadas para ello. El Machine Learning se clasifica generalmente en tres tipos [29]:

- **Aprendizaje supervisado:** El modelo se entrena con datos etiquetados, es decir, con ejemplos donde la entrada y la salida deseada están definidas. El objetivo es que el modelo aprenda una función que mapee correctamente las entradas a las salidas.
- **Aprendizaje no supervisado:** En este caso, el modelo trabaja con datos no etiquetados. El sistema intenta identificar patrones, agrupaciones o estructuras ocultas en los datos sin una salida específica.
- **Aprendizaje por refuerzo:** Este enfoque implica un agente que interactúa con su entorno y aprende a maximizar una recompensa a través de la prueba y error.

4.2.3. Deep Learning

El Deep Leaning (DL) o aprendizaje profundo es una subárea dentro del Machine Learning que se basa en redes neuronales artificiales con múltiples capas de procesamiento. Estas redes, conocidas como redes neuronales profundas, imitan el funcionamiento del cerebro humano, donde las neuronas están conectadas en capas para extraer características complejas de los datos [30]. Lo que distingue al Deep Learning de otros métodos de Machine Learning es su capacidad para trabajar con grandes volúmenes de datos no estructurados, como imágenes, videos y audio. A través de estas redes profundas, el modelo puede aprender representaciones jerárquicas de los datos, donde las capas más cercanas a la entrada

capturan características simples (como bordes en una imagen), mientras que las capas más profundas capturan características abstractas más complejas (como formas de objetos).

Las redes neuronales profundas [31] son las responsables del éxito de muchas aplicaciones modernas de IA, como los sistemas de reconocimiento de imágenes y voz, la conducción autónoma, y los sistemas de recomendación. Sin embargo, entrenar modelos de Deep Learning requiere grandes cantidades de datos y poder computacional significativo, lo que ha llevado al uso de modelos preentrenados y técnicas como la transferencia de aprendizaje, que permite aprovechar modelos ya entrenados para nuevas tareas con menos datos [2].

4.2.4. Visión por Computadora

La Visión por Computadora es un campo de la inteligencia artificial que se centra en desarrollar algoritmos y modelos [32] que permiten a las máquinas interpretar y entender el contenido de imágenes y videos, emulando la capacidad de percepción visual humana [33]. Este proceso incluye tareas como la identificación de objetos, el reconocimiento de patrones, la segmentación de imágenes y la clasificación de objetos, entre otros [34]. Mediante técnicas de Machine Learning, y especialmente redes neuronales convolucionales (CNN), la visión por computadora ha logrado grandes avances, permitiendo la creación de sistemas automatizados para tareas como el reconocimiento facial, la conducción autónoma, etc.

4.2.5. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales (CNN) son un tipo especializado de redes neuronales diseñadas para procesar y analizar datos que tienen una estructura en forma de rejilla, como imágenes o señales de audio [35]. Las CNN se destacan por su capacidad para extraer automáticamente características relevantes de los datos mediante el uso de capas convolucionales, que aplican filtros para detectar patrones locales como bordes, texturas o formas [36]. Estas redes utilizan técnicas como el pooling y activaciones no lineales para reducir la dimensionalidad de los datos y preservar las características más importantes [37], permitiendo una mayor eficiencia en tareas de clasificación y reconocimiento.

4.2.6. ResNet-50

ResNet-50 es una red neuronal convolucional profunda compuesta por 50 capas, diseñada específicamente para superar el problema de la degradación del rendimiento en redes muy profundas, fenómeno en el cual la precisión de los modelos empieza a disminuir a medida que se añaden más capas [38]. Esta arquitectura las llamadas conexiones residuales, que permiten a las capas posteriores recibir la salida directa de capas anteriores, creando atajos que evitan la pérdida de información crítica. Gracias a estas conexiones, ResNet-50 facilita la capacitación de redes significativamente más profundas sin enfrentar problemas comunes como el vanishing gradient (gradientes desaparecientes), donde las actualizaciones de los pesos se vuelven insignificantes durante el entrenamiento [39]. En la **Figura 4** se

muestra cada componente que forma la arquitectura del modelo ResNet-50, el cual incluye [40]:

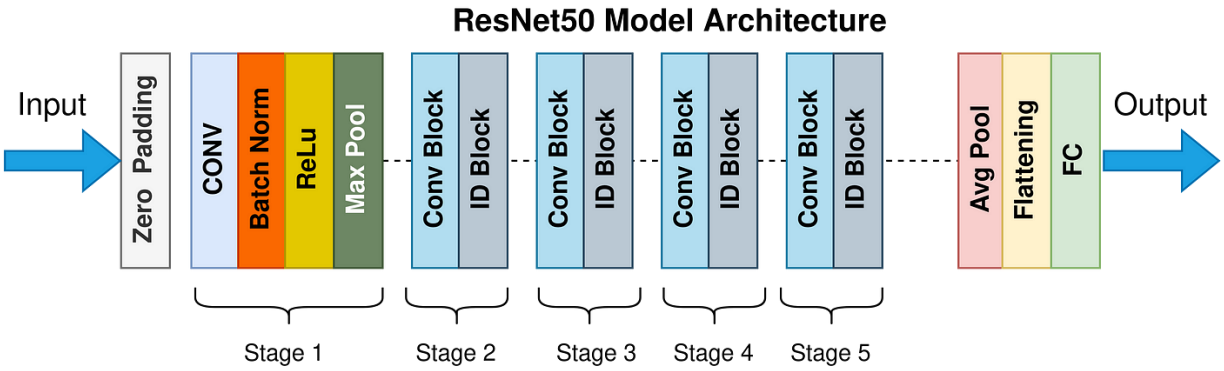


Figura 4: Arquitectura del modelo ResNet-50 *[41]*

1. **Entrada (Input):** El modelo comienza recibiendo la imagen de entrada o, en este caso, el espectrograma de una vocalización de ave. Las dimensiones de entrada típicas para ResNet-50 suelen ser de 224x224 píxeles con 3 canales (RGB).
2. **Zero Padding:** Esta capa añade píxeles adicionales alrededor de los bordes de la imagen para evitar la pérdida de información en las primeras capas convolucionales. Este paso asegura que las dimensiones de la imagen no se reduzcan demasiado rápido.
3. **Capa Convolucional (Conv Layer)**: La primera capa convolucional realiza una operación de convolución sobre la imagen para extraer características iniciales como bordes, texturas y colores. Se utilizan filtros de 7x7 con un stride de 2, lo que reduce la resolución de la imagen, pero capta características importantes.
4. **Normalización por Lotes (Batch Normalization)**: Después de la convolución, se aplica la normalización por lotes, que ajusta y escala las activaciones de la red para estabilizar el proceso de aprendizaje. Esto mejora la convergencia del modelo y reduce la dependencia de inicialización de pesos.
5. **Función de Activación ReLU (Rectified Linear Unit)**: A continuación, se usa una función de activación ReLU, que introduce no linealidades en el modelo y permite que la red aprenda relaciones complejas. Esta función transforma los valores negativos a cero y deja pasar los valores positivos sin cambios.
6. **Max Pooling**: Después de la activación, se aplica una capa de max pooling, que reduce la resolución espacial de la imagen (submuestreo) al seleccionar los valores máximos en regiones 2x2. Esto ayuda a disminuir la cantidad de parámetros y a hacer que la red sea más eficiente.
7. **Bloques Residuales (Residual Blocks)**: El corazón de ResNet-50 está compuesto por varios bloques residuales. Cada uno de estos bloques contiene capas convolucionales que aprenden características a distintos niveles de profundidad. Las

conexiones residuales permiten saltar ciertas capas y pasar directamente la salida de una capa a capas posteriores, preservando información importante y ayudando a evitar el problema de los gradientes desaparecientes.

- **Etapa 2 (Stage 2):** Se compone de 3 bloques residuales.
- **Etapa 3 (Stage 3):** Aquí se encuentran 4 bloques residuales.
- **Etapa 4 (Stage 4)**: Se incrementa la profundidad con 6 bloques residuales.
- **Etapa 5 (Stage 5)**: Finalmente, se utilizan 3 bloques residuales en esta etapa.

8. **Convolución y Pooling en los Bloques Residuales**: Cada bloque residual contiene una serie de capas convolucionales, seguidas por una normalización por lotes y una activación ReLU. Además, en algunos bloques se incluye un max pooling para reducir las dimensiones de las características aprendidas y prepararlas para la siguiente etapa.
9. **Agrupación Promedio Global (Global Average Pooling)**: Esta capa realiza un promedio global sobre todas las características de salida de la última capa convolucional. Reduce drásticamente la dimensionalidad al condensar toda la información en un único vector de características por canal.
10. **Aplanamiento (Flattening)**: El vector resultante de la capa de agrupación promedio es aplanado, lo que significa que se convierte en una matriz de una sola dimensión que puede ser procesada por la capa completamente conectada.
11. **Capa Completamente Conectada (Fully Connected Layer, FC)**: En la última etapa, se aplica una capa completamente conectada que toma las características aprendidas por la red y realiza la clasificación final. En mi proyecto, esta capa realiza la clasificación en las categorías correspondientes a las especies de aves.
12. **Función de Activación Softmax (Softmax Activation)**: La salida de la capa completamente conectada pasa por una función de activación softmax, que convierte los valores en probabilidades y asigna la clase final al espectrograma de entrada, es decir, cuál especie de ave corresponde a la vocalización.

ResNet-50 ha demostrado ser altamente efectiva en tareas de clasificación de imágenes y se ha convertido en una arquitectura ampliamente utilizada en aplicaciones de visión por computadora, debido a su capacidad para aprender representaciones complejas y lograr una alta precisión sin aumentar exponencialmente la complejidad computacional [8].

4.2.7. Transfer Learning

El Transfer Learning o transferencia de aprendizaje es un paradigma de aprendizaje automático que reutiliza modelos preentrenados, originalmente entrenados en grandes conjuntos de datos (como imágenes, texto o voz), para adaptarlos a nuevas tareas o problemas con datos limitados [42]. En lugar de inicializar los pesos de una red neuronal de manera aleatoria, se parte de pesos previamente optimizados y se reentrenan selectivamente

algunas de las capas finales con los datos específicos del nuevo problema [43]. Esta técnica permite transferir el conocimiento adquirido en la tarea original, mejorando tanto la velocidad de entrenamiento como la precisión del modelo en la nueva tarea.

4.2.8. Hiperparámetros para el Ajuste del Modelo

4.2.8.1. Tasa de aprendizaje.

La tasa de aprendizaje (learning rate) es uno de los hiperparámetros más importantes en la optimización de redes neuronales, ya que determina el tamaño de los pasos en los que el modelo ajusta sus pesos [30]. Una tasa muy alta puede provocar una convergencia rápida pero inexacta, mientras que una tasa muy baja puede hacer que el proceso de aprendizaje sea lento o estancado.

4.2.8.2. Épocas (Epoch).

Este hiperparámetro indica cuántas veces el modelo entrenará con el dataset completo. Es importante encontrar un equilibrio para evitar el sobreajuste (cuando el modelo aprende demasiado bien los detalles del dataset de entrenamiento) o el subajuste (cuando no aprende lo suficiente) [44].

4.2.8.3. Tamaño de lote (Batch Size).

El tamaño del lote es el número de muestras de datos que se procesan antes de actualizar los pesos del modelo [45]. Un tamaño de lote pequeño puede hacer que el modelo sea más preciso, pero más lento, mientras que un tamaño de lote grande puede hacer que el modelo sea menos preciso, pero más rápido.

4.2.9. Metodología para Proyectos de Machine Learning CRISP-ML(Q)

CRISP-ML(Q) es una adaptación de la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), orientada específicamente a proyectos de aprendizaje automático (Machine Learning), incorporando un ciclo de retroalimentación y aseguramiento de calidad. Esta metodología es ideal para guiar proyectos de Machine Learning de manera estructurada y sistemática, asegurando tanto la correcta implementación técnica como el control de calidad en cada etapa del desarrollo del modelo [46]. En la **Figura 5** se pueden visualizar las tres secciones principales del ciclo de vida que enmarcan las seis fases correspondientes a la metodología CRISP-ML(Q), imagen tomada de *ml-ops.org*.

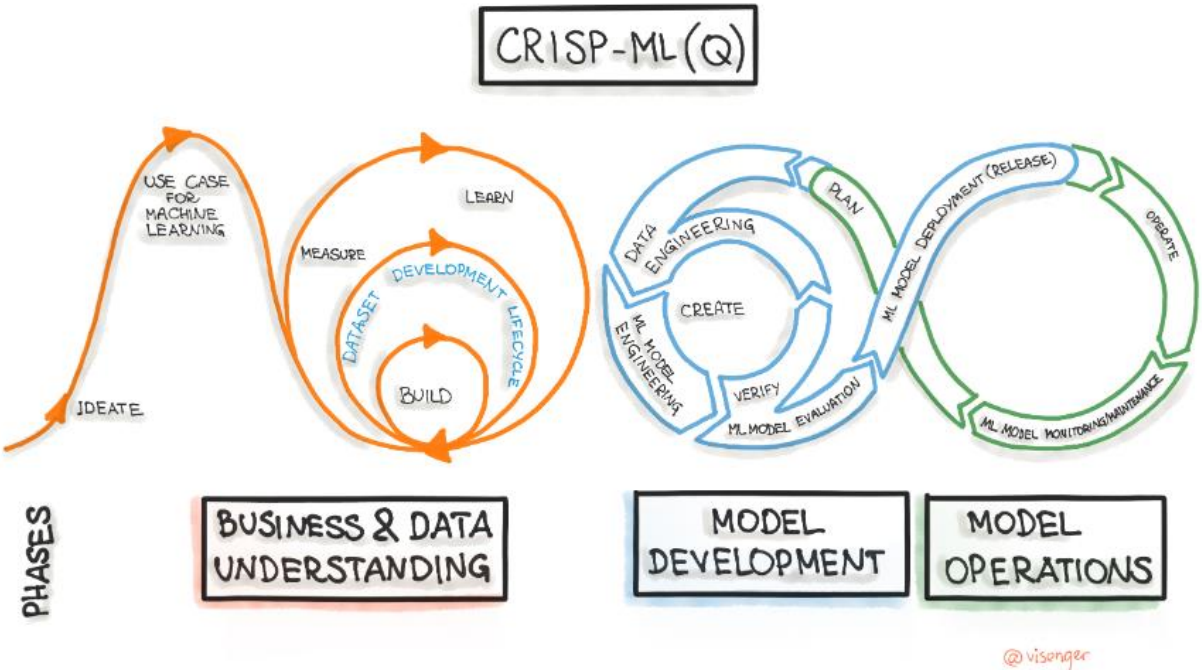


Figura 5: Ciclo de vida de la metodología CRISP-ML(Q) *[46]*

1. **Business & Data Understanding (Comprensión del Negocio y los Datos)**

   En esta primera fase, se define el problema y se comprende el contexto en el cual se aplicará el modelo. Esto incluye:

   - **Identificación del caso de uso:** Clarificar qué se espera lograr con el aprendizaje automático. Por ejemplo, clasificar vocalizaciones de aves.
   - **Recolección y análisis de datos**: Determinar qué datos son necesarios y cómo se obtendrán, así como entender la naturaleza de los mismos.
   - **Definición de objetivos:** Establecer métricas que se utilizarán para evaluar el éxito del modelo.

2. **Ingeniería de Datos**

   Esta fase se enfoca en la preparación y transformación de los datos para el modelado. Incluye:

   - **Limpieza de datos**: Identificación y corrección de errores o inconsistencias en los datos.
   - **Transformación de datos**: Conversión de los datos crudos en un formato adecuado para el análisis, como la generación de espectrogramas a partir de grabaciones de vocalizaciones.
   - **Creación de características**: Desarrollo de nuevas variables que puedan mejorar el rendimiento del modelo.

3. **Ingeniería de Modelos**

   Durante esta fase, se desarrollan y ajustan los modelos de aprendizaje automático. Las actividades incluyen:

   - **Selección de algoritmos:** Elegir el tipo de modelo adecuado, como la red neuronal ResNet-50.
   - **Entrenamiento del modelo:** Aplicar los datos de entrenamiento para que el modelo aprenda a realizar la tarea específica.
   - **Ajuste de hiperparámetros:** Optimización de los parámetros del modelo para mejorar su rendimiento.

4. **Evaluación del Modelo**

   En esta fase, se evalúa el modelo para asegurar que cumple con los objetivos establecidos. Se incluyen:

   - **Pruebas del modelo:** Utilizar un conjunto de datos de prueba para medir el rendimiento del modelo.
   - **Validación:** Comprobar que el modelo no solo se ajusta a los datos de entrenamiento, sino que también generaliza bien a datos no vistos.
   - **Análisis de resultados:** Interpretar las métricas de evaluación y realizar ajustes si es necesario.

5. **Implementación o Despliegue**

   Esta fase se refiere a la puesta en producción del modelo. Implica:

   - **Despliegue**: Implementar el modelo en un entorno real donde pueda hacer predicciones en tiempo real.
   - **Integración**: Asegurar que el modelo se integre adecuadamente con otros sistemas y flujos de trabajo existentes.

6. **Mantenimiento**

   La última fase abarca el monitoreo y la actualización continua del modelo. Esto incluye:

   - **Monitoreo del rendimiento**: Revisar el desempeño del modelo en el tiempo y detectar cualquier degradación.
   - **Actualizaciones**: Ajustar el modelo según sea necesario, ya sea por cambios en los datos de entrada o en las condiciones del entorno.
   - **Reentrenamiento**: Reentrenar el modelo con nuevos datos para mantener su relevancia y precisión.

Un aspecto clave de CRISP-ML(Q) es el ciclo continuo de retroalimentación y mejora. A medida que se avanza a través de cada fase, hay un constante flujo de validación y aprendizaje que asegura que las soluciones proporcionadas cumplen con los estándares de calidad exigidos. Esto incluye la repetición de fases anteriores si es necesario para refinar el modelo o los datos.

4.2.10. Métricas de Evaluación en Clasificación

La evaluación del rendimiento de un modelo de clasificación es esencial para comprender su efectividad y precisión en la identificación de diferentes especies de aves. Para ello, se utilizan varias métricas que ofrecen diferentes perspectivas sobre el rendimiento del modelo.

4.2.10.1. Matriz de Confusión.

La matriz de confusión es una representación visual que permite evaluar el rendimiento de un modelo de clasificación [47]. Proporciona información sobre los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos, organizados en una tabla. Esta matriz facilita la identificación de patrones de errores en la clasificación y permite ver no solo la tasa de aciertos, sino también dónde el modelo está cometiendo errores [48]. En la **Figura 6** se muestra la matriz de confusión con cada una de sus variables correspondientes, imagen tomada de datacamp.com.

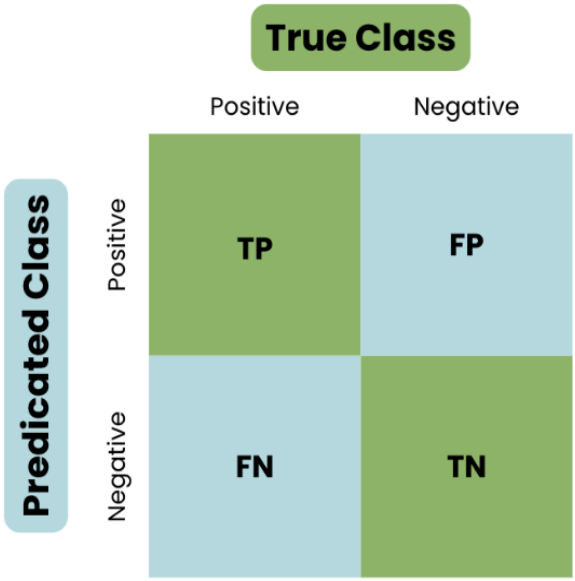


Figura 6: Matriz de confusión con sus respectivos componentes

Se describe cada una de las partes que conforman la matriz de confusión, como [47]:

- **TP (True Positives - Verdaderos Positivos)**: Se refiere a los casos en los que el modelo predijo correctamente la clase positiva. Es decir, el modelo identificó correctamente una especie de ave que realmente era esa especie.
- **FN (False Negatives - Falsos Negativos)**: Se refiere a los casos en los que el modelo no logró identificar correctamente la clase positiva. En este caso, sería cuando el modelo no reconoció una especie de ave que realmente era positiva
- **FP (False Positives - Falsos Positivos)**: Se refiere a los casos en los que el modelo clasificó incorrectamente un ejemplo como positivo cuando en realidad era negativo. En este contexto, significaría que el modelo identificó erróneamente una especie de ave que no era la que se estaba buscando

- **TN (True Negatives - Verdaderos Negativos)**: Se refiere a los casos en los que el modelo predijo correctamente la clase negativa. Es decir, el modelo identificó correctamente una especie de ave que no era la especie objetivo

4.2.10.2. Exactitud (Acurracy).

La exactitud se refiere a la proporción de predicciones correctas realizadas por el modelo en relación con el total de predicciones. Se define como el número de verdaderos positivos (TP) y verdaderos negativos (TN) dividido entre el total de predicciones realizadas (la suma de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). En la **Ecuación 1** se presenta la fórmula para calcular la exactitud:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN}$$

Ecuación 1: Fórmula para calcular la exactitud *[49]*

Es una métrica útil cuando las clases están equilibradas, pero puede ser engañosa si hay un desequilibrio significativo entre las clases, ya que puede ofrecer una alta exactitud incluso si el modelo falla en clasificar correctamente la clase minoritaria.

4.2.10.3. Precisión.

La precisión es una métrica que mide la proporción de verdaderos positivos (TP) entre todos los ejemplos que fueron clasificados como positivos (FP) por el modelo, es decir, el porcentaje de predicciones correctas dentro de todas las predicciones positivas realizadas. En la **Ecuación 2** se presenta la fórmula para calcular la precisión:

$$Precisión = \frac{TP}{TP + FP}$$

Ecuación 2: Fórmula para calcular la precisión *[49]*

Esta métrica es útil cuando se desea conocer cuántas de las especies clasificadas como positivas son realmente correctas.

4.2.10.4. Sensibilidad (Recall).

La sensibilidad, mide la proporción de verdaderos positivos (TP) que fueron correctamente identificados entre todos los ejemplos que realmente pertenecen a la clase positiva. Evalúa la capacidad del modelo para encontrar todos los positivos reales en el conjunto de datos. En la **Ecuación 3** se presenta la fórmula para calcular la sensibilidad:

$$Sensibilidad = \frac{TP}{TP + FN}$$

Ecuación 3: Fórmula para calcular la sensibilidad *[50]*

La sensibilidad es particularmente importante en problemas donde se desea minimizar los falsos negativos, como en el caso de identificar especies en peligro de extinción.

4.2.10.5. F1-Score

El F1-Score es una métrica que combina la precisión y la sensibilidad en un solo valor, siendo útil cuando se busca un equilibrio entre ambas. En la **Ecuación 4** se presenta la fórmula para calcular el F1-Score, esta se calcula como:

$$F1 - Score = 2 \times \frac{Precisión \times Sensibilidad}{Precisión + Sensibilidad}$$

Ecuación 4: Fórmula para calcular el F1-Score *[51]*

Esta métrica combinada es especialmente útil en contextos donde las clases están desbalanceadas.

4.2.11. Audio Digital

El audio digital es una representación del sonido en formato numérico, donde las ondas sonoras, que originalmente son señales continuas, se convierten en datos discretos mediante un proceso conocido como digitalización [52]. Este proceso transforma las variaciones de la onda sonora (que es análoga) en una serie de números binarios que pueden ser procesados, almacenados y transmitidos por dispositivos digitales.

4.2.11.1. Resolución de bits.

Es la cantidad de bits utilizados para representar cada muestra de audio. Una mayor resolución de bits permite una representación más precisa del sonido, capturando detalles más finos [53].

4.2.11.2. Tasa de muestreo.

La tasa de muestreo (sampling rate) es el número de muestras que se toman por segundo de una señal de audio continua para convertirla en una señal discreta (digital) [54]. Se mide en hercios (Hz) y está directamente relacionada con la calidad del audio original.

La tasa de muestreo influye en la fidelidad del audio. Según el teorema de Nyquist, para capturar adecuadamente todas las frecuencias de una señal, la tasa de muestreo debe ser al menos el doble de la frecuencia más alta que se quiera representar [55]. Por ejemplo, la audición humana puede percibir frecuencias de hasta 20,000 Hz, por lo que, para capturar toda la gama audible, se recomienda una tasa de muestreo de al menos 40,000 Hz.

4.2.11.3. Formato de Audio.

Se trabajó con dos de los formatos de audio más comunes: MP3 y WAV. La elección de estos formatos fue determinante para garantizar tanto la calidad de las grabaciones como la compatibilidad con las bibliotecas de procesamiento de audio utilizadas, como librosa y FastAI.

4.2.11.4. Formato MP3.

El MP3 (MPEG-1 Audio Layer III) es un formato de audio comprimido que utiliza compresión con pérdida [56] para reducir significativamente el tamaño del archivo, eliminando las partes del sonido que son menos perceptibles para el oído humano [57]. Este formato es

ideal para almacenar grandes cantidades de datos acústicos debido a su eficiencia en el uso del espacio de almacenamiento. Sin embargo, la compresión puede introducir pequeñas pérdidas de calidad, lo que debe ser considerado al trabajar con grabaciones que requieren precisión acústica.

4.2.11.5. Formato WAV.

El WAV (Waveform Audio File Format) es un formato de audio sin compresión que preserva la calidad original de la grabación [58]. Al no eliminar información del archivo de audio, el formato WAV suele ser considerablemente más grandes que los archivos MP3, pero contienen toda la información del sonido [59].

4.3. Técnicas y Tecnologías

Para el desarrollo e implementación del modelo de clasificación de vocalizaciones de aves se emplearon diversas técnicas de recopilación de audios, herramientas y librerías de software especializadas, tanto para la preparación de los datos como para la construcción, entrenamiento y evaluación de los algoritmos de Machine Learning.

4.3.1. Técnicas de Recopilación de Datos

En el ámbito de la clasificación acústica de especies de aves, la recopilación de datos juega un papel fundamental. Para garantizar la calidad y relevancia del conjunto de datos utilizado, se emplean diferentes técnicas que permiten obtener grabaciones de manera sistemática y precisa. En el presente estudio, el método principal para la recolección de datos es el web scraping, complementado con consideraciones éticas y legales que aseguran un uso responsable de los recursos disponibles.

4.3.1.1. Web Scraping.

El web scraping es una técnica automatizada que permite extraer información de sitios web, mediante el uso de scripts o programas diseñados para recolectar datos específicos de una o más páginas web [60]. En este caso, el proceso se enfoca en extraer grabaciones de audio de aves desde la plataforma Xeno-canto, que contiene un amplio repositorio de sonidos de especies alrededor del mundo.

Las grabaciones extraídas deben ser filtradas por especie, ubicación geográfica y calidad de sonido, asegurando que solo se utilicen aquellas que cumplen con los criterios definidos para el estudio.

4.3.1.2. Consideraciones éticas y legales.

Aunque el web scraping es una técnica poderosa, es fundamental tomar en cuenta las consideraciones éticas y legales al realizar la recolección de datos de sitios web. Muchas plataformas, como Xeno-canto, establecen condiciones de uso específicas que deben ser respetadas. Entre ellas, puede incluirse la limitación en la cantidad de datos que se pueden descargar, así como el requerimiento de citar y atribuir adecuadamente las grabaciones a los autores o a la plataforma misma.

4.3.1.3. Licencias de Uso: Creative Commons Attribution-NonCommercial (CC BY-NC).

Plataformas como Xeno-canto están sujetas a licencias Creative Commons Attribution-NonCommercial (CC BY-NC). Este tipo de licencia permite a los usuarios compartir, copiar y redistribuir el material en cualquier medio o formato, así como adaptar o modificar la obra para propósitos no comerciales [61], siempre y cuando se cumplan dos condiciones fundamentales [62]:

1. **Atribución:** Es obligatorio dar crédito al autor original de la grabación de manera adecuada, proporcionando su nombre y la fuente de la obra, sin sugerir que el autor original apoya el uso de su material en este proyecto.
2. **Uso No Comercial:** Las grabaciones no pueden ser utilizadas con fines comerciales, es decir, no pueden generar ingresos directos ni indirectos a partir de su uso.

4.3.2. Técnicas de Preprocesamiento de Audio

En la clasificación acústica de especies de aves, el preprocesamiento de señales de audio es un paso necesario para asegurar que los datos utilizados sean adecuados para su análisis y clasificación. Estas técnicas permiten mejorar la calidad de las grabaciones al eliminar ruidos, normalizar los niveles de sonido y segmentar las señales de interés. Además, el uso de espectrogramas facilita la transformación de las señales de audio en representaciones visuales, lo que posibilita el empleo de técnicas avanzadas de aprendizaje automático para la identificación de patrones acústicos.

4.3.2.1. Procesamiento de Señales.

El preprocesamiento de señales es un área de la ingeniería y la ciencia de la computación que se ocupa de la representación, manipulación y transformación de señales que contienen información auditiva [63]. Hace referencia al conjunto de técnicas y transformaciones que se aplican a datos en bruto, como grabaciones de audio, video o sensores, para convertirlas en representaciones mejor estructuradas que resalten la información de interés para análisis posteriores [64]. Operaciones típicas incluyen filtrado para reducción de ruido, segmentación, transformadas (Fourier, Wavelet) para análisis frecuencial, escalado de amplitudes, entre otras [65]. El preprocesamiento busca “limpiar” y estandarizar los datos de entrada para facilitar la extracción eficiente de características distintivas mediante algoritmos especializados.

4.3.2.2. Espectrograma de Audio.

Un espectrograma de audio es una representación visual de las frecuencias que componen una señal acústica a lo largo del tiempo [66]. Gráficamente muestra en el eje X el tiempo, en el eje Y las frecuencias desde graves a agudas, y mediante una escala de colores o una tercera dimensión indica la amplitud o intensidad de cada frecuencia en cada instante [67]. Esta imagen espectral permite analizar cambios en timbres, identificar fuentes sonoras,

ver modulaciones, y extraer otra información relevante de la señal de audios. Los espectrogramas se generan típicamente utilizando la Transformada de Fourier de Tiempo Corto (STFT) [68], que divide la señal de audio en segmentos cortos y calcula la transformada de Fourier para cada segmento. En la **Figura 7** se presenta un ejemplo de representación desde la forma original de onda a espectrograma con escala lineal.

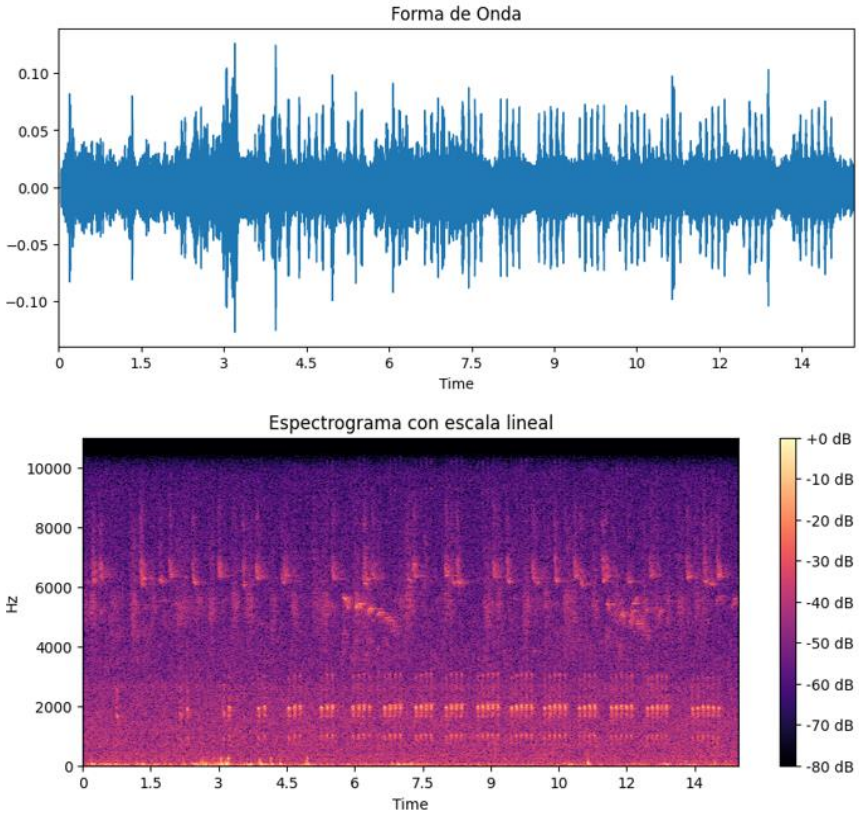


Figura 7: Ejemplo representación de forma de onda a espectrograma

Dentro del análisis de sonidos de aves, los espectrogramas son particularmente útiles porque permiten visualizar características importantes de las vocalizaciones, como [33]:

1. **Frecuencia fundamental:** La frecuencia base de la vocalización.
2. **Armónicos:** Múltiples enteros de la frecuencia fundamental que dan riqueza al sonido.
3. **Modulación de frecuencia:** Cambios en la frecuencia a lo largo del tiempo, comunes en los cantos de aves.
4. **Duración:** La longitud temporal de diferentes elementos del canto.
5. **Intensidad:** Representada por la intensidad del color del espectrograma.

Con toda esta información auditiva en una representación visual que puede ser procesada por algoritmos de visión por computadora como las CNN. Preservan información temporal y frecuencial simultáneamente, capturando la estructura compleja de las vocalizaciones de las aves.

4.3.2.3. Transformaciones de Audio.

Las transformaciones de audio son técnicas aplicadas a señales de audio para modificar o mejorar sus características [69], generalmente con el objetivo de mejorar el rendimiento de modelos de machine learning o deep learning. Estas transformaciones permiten aumentar el tamaño del conjunto de datos y a la vez, mejorar la capacidad de generalización del modelo al exponerlo a variaciones del audio original.

Principales Transformaciones de Audio

1. **Redimensionado de señal (ResizeSignal):** Esta transformación ajusta la longitud de la señal de audio a una duración predefinida [70]. En este proyecto, se emplea la técnica de padding, repitiendo el audio original hasta alcanzar la longitud deseada (en este caso, 15 segundos). Este paso es importante para garantizar que todas las muestras tengan la misma longitud al momento de ser procesadas por el modelo.
2. **Añadir ruido (AddNoise):** Esta técnica introduce pequeñas cantidades de ruido en la señal de audio, simulando la presencia de sonidos ambientales [71]. El propósito es hacer que el modelo sea más robusto frente a variaciones en las condiciones del entorno, lo que mejora su capacidad de generalización.
3. **Normalización (Normalize):** Esta transformación ajusta los valores de amplitud del audio para que estén en una escala común, generalmente entre 0 y 1 o normalizando según la media y la desviación estándar de la señal [72]. La normalización es clave para evitar que las diferencias en volumen o intensidad afecten al rendimiento del modelo.

4.3.3. Técnicas de Aumento de Datos

El aumento de datos es esencial en el aprendizaje profundo, ya que permite ampliar el conjunto de datos y prevenir el sobreajuste (ovefitting) [73]. Las siguientes técnicas se aplican a nivel de lote, una vez que las muestras ya han sido procesadas.

1. **Cambio de Volumen (ChangeVolume):** Modifica aleatoriamente el volumen de las señales de audio, exponiendo al modelo a distintas intensidades y ayudándolo a adaptarse a diferentes condiciones acústicas [74].
2. **Máscara Temporal (TimeMask):** Oculta o enmascara aleatoriamente secciones de tiempo en el espectrograma [75]. Esto obliga al modelo a aprender características globales de audio sin depender de una secuencia temporal específica.
3. **Máscara de Frecuencia (FrequencyMask):** Similar al TimeMask, esta técnica oculta bandas de frecuencia aleatorias del espectrograma. La idea es que el modelo aprenda a identificar patrones en las frecuencias que no dependen solo de un rango específico [75].

4.3.4. Técnicas de Optimización y Entrenamiento de Modelo

En el entrenamiento de modelos de aprendizaje profundo, es imprescindible utilizar estrategias de optimización que mejoren la precisión del modelo, aumenten su capacidad de generalización y eviten el sobreajuste [76]. Para ello, se emplean varias técnicas que ajustan automáticamente los parámetros del modelo durante el entrenamiento. A continuación, se describen algunas de las más importantes aplicadas en este proyecto.

1. **Regularización:** Es una técnica clave utilizada para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. En este proyecto, se aplica la decadencia de pesos (weight decay), una forma de regularización que penaliza los pesos grandes en los parámetros del modelo [77]. La regularización es efectiva porque actúa como una forma de suavizado, lo que permite que el modelo se enfoque en las características más importantes de los datos sin aprender patrones específicos del conjunto de entrenamiento que no se generalizan bien en nuevos datos.
2. **Callbacks:** Son herramientas que permiten intervenir en momentos clave del proceso de entrenamiento, facilitando el ajuste de parámetros, el guardado de modelos y otras tareas relacionadas con la optimización [78]. En este proyecto, se han utilizado tres callbacks principales:
   - **SaveModelCallback:** Este callback guarda el mejor modelo obtenido durante el entrenamiento, en función de una métrica de monitorización específica (en este caso es la precisión) [79]. De esta manera, el modelo que se retiene al final es el que se ha alcanzado el mejor rendimiento, lo que garantiza que no se pierde la mejor configuración de pesos obtenida.
   - **ReduceLROnPlateau:** Esta se encarga de ajustar dinámicamente la tasa de aprendizaje. Este mecanismo reduce la tasa de aprendizaje automáticamente cuando el rendimiento del modelo en el conjunto de validación no mejora después de un cierto número de épocas [79]. El propósito es ralentizar el entrenamiento para que el modelo pueda refinar sus ajustes en las etapas finales, acercándose al mínimo óptimo sin desestabilizar el proceso.
   - **EarlyStoppingCallback:** Detiene el entrenamiento cuando la mejora en el conjunto de validación se estabiliza, es decir, cuando no hay mejoras significativas en la pérdida de validación durante un número determinado de épocas consecutivas [79]. Su propósito es evitar el sobreentrenamiento (overtraining), que puede llevar al sobreajuste del modelo.

4.3.5. Lenguaje de Programación Python

Python es uno de los lenguajes de programación más utilizados en el campo de la inteligencia artificial y el machine learning debido a su simplicidad, versatilidad y una amplia comunidad de desarrollo [80]. Su ecosistema cuenta con bibliotecas especializadas como

TensorFlow, PyTorch y FastAI, que facilitan la implementación de modelos de aprendizaje profundo, así como otras herramientas como NumPy y Pandas [81] para el manejo de datos. Python permite desarrollar prototipos de manera eficiente y ha sido clave en la evolución de proyectos de análisis de datos y modelado predictivo gracias a su facilidad para integrar algoritmos complejos.

4.3.6. Librerías de Python

4.3.6.1. Pandas.

Pandas es una biblioteca esencial para la manipulación y análisis de datos, particularmente útil para trabajar con datos estructurados y tabulares como los obtenidos de archivos CSV, bases de datos, o archivos Excel [82]. Ofrece dos estructuras de datos principales: DataFrame, para manejar tablas de datos en formato de filas y columnas, y Series, para manejar arreglos unidimensionales. Pandas permite realizar operaciones como filtrado, agrupamiento, limpieza y transformación de datos de manera eficiente [83]. Además, facilita la integración con otras bibliotecas populares como NumPy para cálculos numéricos y Matplotlib o Seaborn para la visualización de datos, haciendo que sea una herramienta clave para el análisis exploratorio de datos y la preparación de datasets para machine learning.

4.3.6.2. Request.

Request es una biblioteca ampliamente utilizada para hacer solicitudes HTTP de manera simple y eficiente. Permite interactuar con APIs y descargar datos a través de peticiones como GET y POST, entre otras, manejando automáticamente aspectos complejos como la autenticación y las cookies [84]. En este estudio, Requests es esencial para conectarse a la API de Xeno-canto, permitiendo descargar los metadatos y grabaciones de las aves filtradas por especie o región. La biblioteca también maneja formatos de respuesta comunes como JSON y XML, lo que facilita la manipulación de los datos obtenidos para su posterior análisis o almacenamiento.

4.3.6.3. Tqdm.

Tqdm es una biblioteca utilizada para crear barras de progreso en tiempo real, lo que facilita el seguimiento de la ejecución de bucles o procesos largos, como descargas, transformaciones de datos, o entrenamientos de modelos en machine learning [85]. Su implementación es sencilla y se integra fácilmente con Pandas, NumPy, y otros módulos, proporcionando retroalimentación visual que mejora la eficiencia del desarrollo y depuración.

4.3.6.4. Pydub.

Pydub es una biblioteca especializada en la manipulación y procesamiento de archivos de audio. Además de permitir la conversión entre formatos como MP3, WAV, FLAC, y otros. Pydub ofrece funcionalidades para realizar operaciones como cortar, concatenar, ajustar el volumen, y aplicar efectos como silencios o fundidos [86]. Es especialmente útil en proyectos

que requieren el preprocesamiento de archivos de audio antes de analizarlos o utilizarlos en modelos de machine learning.

4.3.6.5. Librosa.

Librosa es una biblioteca ampliamente utilizada para el análisis y procesamiento de música y audio [87]. Además de la carga y manipulación de archivos de audio, Librosa permite extraer características acústicas esenciales como MFCCs (Mel Frequency Cepstral Coefficients), chroma features, tonnetz, y spectral contrast, que son fundamentales para tareas de clasificación de audio. Librosa también facilita la transformación de señales de audio en espectrogramas y permite aplicar técnicas como la re-sampleación y la detección de silencios [88].

4.3.7. PyTorch

PyTorch es una biblioteca de código abierto ampliamente utilizada para tareas de machine learning y deep learning. PyTorch facilita el manejo eficiente de datos mediante estructuras llamadas tensores, que permiten operaciones de álgebra lineal con aceleración por GPU. Las capas del modelo, como en la arquitectura ResNet-50, están definidas en PyTorch y optimizadas mediante técnicas como backpropagation y gradiente descendente [89]. Además, las optimizaciones, como el ajuste dinámico de la tasa de aprendizaje y regularización, se llevan a cabo en PyTorch, gracias a la integración con FastAI. Esta biblioteca tiene un componente específico para audio denominado Torchaudio facilita la conversión de audios en espectrogramas, la extracción de características como MFCCs (Mel Frequency Cepstral Coefficients) [90], y la aplicación de preprocesamientos esenciales para entrenar modelos. Al estar integrado con PyTorch, permite una fácil interoperabilidad con redes neuronales y flujos de trabajo de deep learning, optimizando proyectos de clasificación y análisis de audio.

4.3.8. FastAI

Es una biblioteca de alto nivel construida sobre PyTorch que facilita el desarrollo rápido de modelos de machine learning y deep learning. Su principal objetivo es hacer que las técnicas avanzadas de aprendizaje profundo sean más accesibles a desarrolladores y no expertos, permitiendo crear modelos de manera eficiente con pocas líneas de código [91]. FastAI ofrece un enfoque simplificado para tareas como clasificación de imágenes, procesamiento de texto, e incluso audio, mediante su API intuitiva. Además, proporciona herramientas de transfer learning, permitiendo a los usuarios aprovechar modelos preentrenados como ResNet-50 para adaptarlos a nuevas tareas.

FastAI está compuesto por una serie de componentes modulares que permiten gestionar datos, configurar modelos y ejecutar procesos de entrenamiento de manera eficiente. Dos de sus componentes que se ocupan son DataBlock y DataLoader, que se encargan del manejo eficiente de datos para alimentar los modelos.

4.3.8.1. DataBlock.

Es una de las herramientas más flexibles y poderosas de FastAI. Este componente define un pipeline de procesamiento de datos, lo que permite especificar cómo deben ser transformados y organizados los datos antes de ser entregados al modelo para su entrenamiento [92].

Un DataBlock permite definir:

- **Tipos de datos:** Imágenes, texto, audio, etc.
- **Métodos de división:** Especificar cómo dividir los datos en entrenamiento y validación.
- **Transformaciones:** Qué preprocesamientos deben aplicarse, como la generación de espectrogramas en audio o la normalización de imágenes.
- **Labelling:** Cómo etiquetar los datos, lo que es indispensable para las tareas de clasificación.

4.3.8.2. DataLoader.

Es el siguiente paso en el pipeline de datos. Una vez que el DataBlock organiza y transforma los datos, el DataLoader se encarga de cargarlos de manera eficiente en lotes (batches) para entrenar el modelo [92].

El DataLoader:

- **Carga los datos en lotes:** Facilita la lectura y procesamiento de grandes cantidades de datos al dividirlos en conjuntos más pequeños.
- **Aplicación de transformaciones en tiempo real:** Puede aplicar transformaciones durante la carga de datos, como rotaciones o cambios de brillo en imágenes o modificaciones en espectrogramas de audio.
- **Soporta múltiples workers:** Permite cargar los datos en paralelo usando varios núcleos del procesador, acelerando significativamente el proceso.

4.3.9. Google Colab

Google Colab es una plataforma en la nube que permite ejecutar código Python, especialmente orientada al desarrollo de proyectos de machine learning y deep learning. Colab proporciona un entorno gratuito con acceso a GPU y TPU, lo que facilita la ejecución de modelos complejos de manera más rápida que en una computadora personal [93]. También permite la colaboración en tiempo real, similar a Google Docs, lo que es útil para equipos de desarrollo. Es compatible con bibliotecas populares como TensorFlow, PyTorch, Fast.ai, y Torchaudio, lo que lo convierte en una herramienta ideal para prototipos y experimentos. Además, ofrece integración directa con Google Drive para almacenar datos y proyectos [56].

4.4. Trabajos Relacionados

Las redes neuronales convolucionales (CNN) han demostrado resultados sobresalientes en tareas actuales de visión artificial y procesamiento automatizado de imágenes y audio para la identificación y clasificación de elementos. Diversos proyectos se han enfocado en aplicar y optimizar estas arquitecturas de machine learning para lograr un preciso reconocimiento de especies animales mediante sus vocalizaciones y atributos visuales. En el campo específico de la ornitología computacional destacan trabajos orientados tanto al reconocimiento general de aves silvestres como a la discriminación de características distintivas en especies endémicas particulares.

El presente trabajo toma como punto de partida estas investigaciones previas, proponiendo un modelo especializado en tres especies locales mediante la técnica de transfer learning sobre redes convolucionales preentrenadas. En la **Tabla 1** se presentan 10 trabajos relacionados para la implementación del trabajo de integración curricular.

Tabla 1: Trabajos relacionados

Código	Título	Referencia	Resumen
TR01	Automated Bird Species Identification using Audio Signal Processing and Neural Network	[94]	Este trabajo presenta un enfoque de clasificación de especies de aves mediante el uso de redes neuronales convolucionales (CNN) aplicadas a grabaciones de audio. Se destaca la efectividad de las CNN en el procesamiento de audio, lo que puede ser un soporte para el uso de ResNet-50 y espectrogramas en la identificación de aves.
TR02	Automated Birdsong Recognition in Complex Acoustic Enviroments: A Review	[95]	Este estudio investiga la identificación de cantos de aves utilizando técnicas de aprendizaje profundo, resaltando la importancia de las características espectrales en el reconocimiento. Esto respalda la metodología al resaltar que las características acústicas son significativas para la clasificación de especies.
TR03	Survey Study on the Methods of Bird Vocalization Classification	[96]	Este trabajo revisa diferentes enfoques y metodologías para la clasificación de especies de aves mediante vocalizaciones. Proporciona una visión general de los métodos tradicionales

Código	Título	Referencia	Resumen
			y modernos, lo que puede ser útil para contextualizar este proyecto dentro del panorama existente de la investigación en este campo.
TR04	Birds Identification System Using Deep Learning	[97]	Este artículo revisa el uso de técnicas de aprendizaje profundo en la identificación de aves, resaltando el potencial de las CNN y el aprendizaje transferido, que se alinea con el enfoque de usar ResNet-50. Además, discute los conjuntos de datos y la importancia de la calidad del audio, que son aspectos clave dentro de esta investigación.
TR05	A Transfer Learning Strategy for Owl Sound Classification by Using Image Classification Model with Audio Spectrograma	[98]	Este trabajo presenta un enfoque innovador para la identificación automática de especies de aves basado en el análisis del paisaje sonoro, utilizando transferencia de aprendizaje para mejorar la precisión con conjuntos de datos limitados. Se exploran técnicas avanzadas para extraer características acústicas clave, como espectrogramas y coeficientes cepstrales de frecuencia de Mel (MFCC), para maximizar el uso de datos limitados y abordar problemas como el sobreajuste. Este enfoque complementa el análisis de vocalizaciones específicas, destacando la utilidad de modelos como ResNet-50 en la clasificación precisa de sonidos de aves, incluso con datos escasos.
TR06	Birdsong Detection at the Edge with Deep Learning	[99]	Este artículo investiga el uso de aprendizaje transferido en la clasificación de vocalizaciones de aves, utilizando CNN. Destaca la efectividad de este enfoque para mejorar la precisión en datasets limitados, lo cual es relevante para

Código	Título	Referencia	Resumen
			este proyecto con especies específicas en una región determinada.
TR07	Bird Species Identification Through Vocalization Analysis Using Machine Learnig	[100]	Este trabajo analiza varias técnicas de machine learning para clasificar vocalizaciones de aves. Compara diferentes tipos de redes neuronales y sus resultados, proporcionando un marco de referencia para elegir el mejor enfoque en la investigación.
TR08	Bird Species Recognition Using Unsupervised Modeling of Individual Vocalization Elements	[101]	Este artículo revisa las aplicaciones de machine learning en el monitoreo de vida silvestre, incluidas las aves. Resalta cómo estos métodos pueden mejorar la identificación y seguimiento de especies, lo que puede ofrecer información valiosa para la aplicación del modelo en la identificación de aves en Loja, Ecuador.
TR09	Bird Call Recognition Using Deep Convolutional Neural Network, ResNet-50	[102]	Este estudio presenta un sistema automático de clasificación de especies de aves a partir de vocalizaciones usando aprendizaje profundo. Se analizan los resultados obtenidos y la eficiencia del modelo, ofreciendo un enfoque directamente comparable con la aplicación de ResNet-50 en la clasificación de especies específicas en esta investigación.
TR10	Data-Efficient Classification of Birdcall Through Convolutional Neural Networks Transfer Learning	[103]	Este estudio también explora el uso de técnicas de aprendizaje profundo, especialmente mediante transferencia de aprendizaje y el uso de espectrogramas como representación visual de los sonidos. La investigación destaca casos exitosos en la identificación de especies mediante la arquitectura ResNet-50, subrayando la adaptabilidad de estos métodos a distintos tipos de datos acústicos. Esto

Código	Título	Referencia	Resumen
			resulta particularmente relevante para la clasificación de especies de aves locales, dada la eficacia de ResNet-50 en la clasificación de vocalizaciones complejas y diversas.

4.4.1. Conclusión sobre los trabajos relacionados

En primer lugar, varios estudios han demostrado la eficacia de las técnicas de aprendizaje profundo, como las redes neuronales convolucionales, para identificar especies a partir de grabaciones de audio, lo que es central en el proyecto (TR01, TR02, TR06, TR07, TR09). Además, estos estudios destacan la importancia de utilizar características espectrales representativas, como los espectrogramas, en el proceso de clasificación (TR02, TR05, TR10) y subrayan la necesidad de datasets especializados y de alta calidad para entrenar modelos efectivos (TR03, TR04, TR06). También se resalta el potencial del aprendizaje transferido para mejorar la precisión en conjuntos de datos limitados, lo cual es especialmente relevante para el estudio de especies en regiones específicas y de difícil acceso (TR05, TR06). Por último, la revisión de enfoques existentes proporciona un marco teórico y práctico que fortalecerá la implementación de la metodología CRISP-ML(Q), asegurando la robustez y efectividad del modelo propuesto para el reconocimiento de las especies Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

5. Metodología

5.1. Área de estudio

El Trabajo de Integración Curricular (TIC) titulado “Transferencia de Aprendizaje Hacia el Modelo ResNet-50 para la Clasificación de Especies de Aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris”. Se realizó en un contexto de estudio de las vocalizaciones de aves que habitan en Ecuador, particularmente en el Parque Nacional Podocarpus, una de las áreas protegidas más importantes del país. Este parque se extiende por las provincias de Loja y Zamora Chinchipe, caracterizándose por su altitud, que varía entre 1,500 y 3,500 metros sobre el nivel del mar, permitiendo una alta biodiversidad y un hábitat ideal para las especies en estudio.

Las coordenadas geográficas de referencia para el Parque Nacional Podocarpus son: **Latitud:** -4.28989 y **Longitud:** -78.98907. La vegetación densa y la alta humedad predominan en esta región, lo cual proporciona un entorno idóneo para la presencia de especies endémicas, como el Churrín Negruzco, el Soterrey Cola Pálida y el Soterrey Montés de Pecho Gris, que son el foco de este estudio.

En la **Figura 8** se presenta un mapa que indica el área geográfica del Parque Nacional Podocarpus, imagen recuperada de parks and tribes [104].

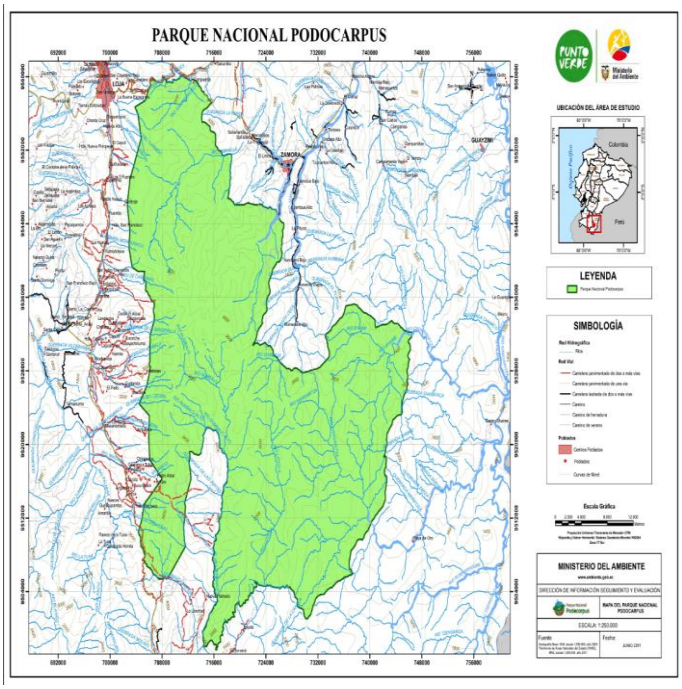


Figura 8: Mapa detallado del Parque Nacional Podocarpus

5.2. Procedimiento

El presente proyecto se desarrolló incorporando de manera integral recursos científicos, técnicos y analíticos, asegurando la validez, precisión y confiabilidad en cada etapa del proceso, desde la recolección de datos hasta la interpretación de los resultados. La metodología destinada para proyectos de Machine Learning, CRISP-ML(Q) (Croos-Industry

Standard Process for Machine Learning), fue seleccionada y adaptada específicamente para responder a las necesidades de este proyecto. La metodología se estructuró en cuatro fases, las tres primeras relacionadas con el cumplimiento del Objetivo 1 y la cuarta fase con el Objetivo 2.

5.2.1. Objetivo 1: Ajustar el modelo ResNet-50 con un conjunto de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

5.2.1.1. Fase 1: Comprensión de datos y negocio.

La primera fase del proyecto se inició con una revisión de literatura en bases de datos especializadas como IEEE Xplore, arXiv, Google Scholar, Scopus y ACM (ver sección **Trabajos Relacionados**), lo que permitió establecer una base sólida de conocimiento y documentar los requerimientos técnicos y funcionales necesarios para el desarrollo del proyecto. Paralelamente, se realizó una entrevista detallada con el Ingeniero Oscar Cumbicus, experto en machine learning, quien proporcionó orientación fundamental sobre los aspectos teóricos y prácticos del proyecto (ver **Anexo 1**). La información recolectada más relevante se resume (ver sección **6.1.1**).

5.2.1.2. Fase 2: Ingeniería de datos.

Tarea 1: Selección de datos

La selección de datos fue guiada por criterios de calidad y representatividad, tales como una tasa de muestreo en 8 kHz y 48 kHz, duración mínima de 5 segundos y ausencia de ruidos significativos, asegurando así una muestra adecuada de las vocalizaciones de las especies. A partir de estos criterios, se implementó un proceso sistemático de web scraping, desarrollado específicamente para extraer grabaciones de la plataforma Xeno-canto. Este proceso se realizó utilizando bibliotecas especializadas como requests, pandas, tqdm, os y time, permitiendo una recolección automatizada y eficiente (ver sección **6.1.2.1**). En la **Figura 9** se muestra la recopilación de grabaciones mediante la técnica de web scraping.

Figura 9: Técnica web scraping desde Xeno-canto a repositorio

Las grabaciones extraídas se almacenaron en un sistema estructurado en Google Drive, garantizando su organización y accesibilidad para las fases posteriores del proyecto.

Tarea 2: Ingeniería de características

El procesamiento de las grabaciones comenzó con la conversión estandarizada de todos los archivos al formato MP3 utilizando la biblioteca pydub. Durante esta fase, se identificaron y eliminaron archivos inválidos que presentaban problemas como descargas incompletas, codificación incorrecta, incompatibilidades, grabaciones defectuosas o metadatos dañados. Un aspecto fundamental fue la estandarización de la tasa de muestreo a 44100 Hz, ya que se identificó como la frecuencia predominante en el conjunto de datos, aplicando técnicas de resampling para asegurar la consistencia (ver sección **6.1.2.2**).

Tarea 3: Estandarización de datos

Para enriquecer el conjunto de datos, se implementaron técnicas de aumento utilizando librosa, shutil, soundfile y sklearn. Las transformaciones incluyeron cambios de tono y adición de ruido controlado, logrando un balance en el dataset con 500 audios por cada especie, para un total de 1500 muestras. Esta colección se dividió estratégicamente, destinando el 90% (1350 muestras) para entrenamiento y el 10% restante (150 muestras) para pruebas. La generación de espectrogramas se realizó utilizando FastAI y Fastaudio, con configuraciones específicas que incluyen una duración de 15 segundos y parámetros n_fft=1024 para la transformada de Fourier (ver sección **6.1.2.3**).

5.2.1.3. Fase 3: Ingeniería de modelos de aprendizaje automático.

Tarea 1: Selección de los hiperparámetros

Los hiperparámetros del modelo se seleccionaron mediante un proceso sistemático. Este proceso comenzó con la configuración de la arquitectura ResNet-50. Se estableció una capa de entrada adaptada para señales de audio (n_in=1) y se implementó la función de pérdida CrossEntropyLossFlat, junto con métricas de accuracy y error_rate para el seguimiento del rendimiento. El proceso incluyó la búsqueda automática de la tasa de aprendizaje óptima para garantizar un entrenamiento eficiente (ver sección **6.1.3.1**).

Tarea 2: Aplicación de Transfer Learning

La implementación del transfer learning, se realizó mediante la configuración específica de callbacks estratégicos. Se implementó SaveModelCallback para el guardado automático del mejor modelo, ReduceLROnPlateau para el ajuste adaptativo de la tasa de aprendizaje y EarlyStoppingCallback para prevenir el sobreajuste. El entrenamiento se ejecutó durante 30 épocas, permitiendo que el modelo se adaptará gradualmente a las características específicas de las vocalizaciones de aves (ver sección **6.1.3.2**).

Tarea 3: Versionamiento de modelos

Se estableció un sistema de control de versiones para el modelo que permitió registrar y documentar cada iteración. Los modelos se guardaron en formato .pth, manteniendo un registro detallado de los cambios implementados y las mejoras obtenidas en cada versión (ver sección **6.1.3.3**).

5.2.2. Objetivo 2: Evaluar el porcentaje global de acierto del modelo entrenado en la clasificación de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

5.2.2.1. Fase 4: Evaluación de modelos de aprendizaje automático.

Tarea 1: Validación del rendimiento del modelo

El proceso de validación se implementó mediante el análisis de la matriz de confusión generada por la versión del mejor modelo, lo que permitió identificar las predicciones más confusas y obtener percepciones valiosas sobre el comportamiento del modelo. A partir de este análisis, se calcularon las métricas de precisión, acierto, tasa de error, sensibilidad y F1-score para evaluar el desempeño. Se utilizó un DataLoader específico para el conjunto de prueba, asegurando la consistencia en el procesamiento de datos y facilitando la validación detallada de cada versión (ver sección **6.2.1.1**).

Tarea 2: Empaquetado del modelo

El empaquetado del modelo incluyó su exportación desde el formato .pth generado durante la aplicación del transfer learning al formato .pkl, junto con toda la documentación necesaria sobre dependencias y requisitos técnicos (ver **Anexo *3***). Se desarrolló un script de predicción que implementa el pipeline completo de procesamiento, desde la carga del audio hasta la generación del espectrograma y la predicción final (ver sección **6.2.1.2**). Además, como una validación adicional, se desplegó el modelo en un sitio web alojado localmente, (ver **Anexo *4***), donde se verificó su capacidad para identificar correctamente la especie de ave según el archivo de audio cargado.

5.3. Recursos e Instrumentos utilizados

5.3.1. Hardware

En la **Tabla 2** se detallan las especificaciones de la laptop empleada para el desarrollo del proyecto.

Tabla 2: Especificaciones de la laptop utilizada en el proyecto

Componente	Especificación
Marca y Modelo	Dell Inspiron 14 5410
Procesador	11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
Memoria RAM	16.0 GB
Sistema Operativo	Windows 11 Pro
Disco Duro	500 GB NVMe
Tarjeta Gráfica	Intel(R) Iris(R) Xe Graphics

5.3.2. Software

Se utilizaron las siguientes herramientas de software para las distintas etapas de recolección, preprocesamiento y modelado:

- **Procesamiento de datos:** Pandas, request, tqdm, os, time, pydub.
- **Procesamiento de audio:** Librosa, soundfile, shutil, pydub.
- **Machine Learning:** FastAI, Fastaudio, PyTorch, Keras, Scikit-learn.
- **Entorno de desarrollo:** Google Colab.
- **Plataformas de datos:** Xeno-canto.
- **Almacenamiento en la nube:** Google Drive.

5.3.3. Instrumentos de recolección de datos

Para la recolección de datos se empleó una entrevista con el Ingeniero Oscar Cumbicus, especialista en machine learning, quien proporcionó asesoramiento sobre la aplicación del transfer learning y la optimización del modelo ResNet-50 para la clasificación de vocalizaciones de aves. Además, se aplicaron técnicas de web scraping mediante Python para recolectar grabaciones de audio desde la plataforma Xeno-canto, en cumplimiento con sus normativas de uso científico.

5.3.4. Consideraciones Bioéticas

El proyecto se adhiere estrictamente a las normativas de uso de datos de biodiversidad establecidas por Xeno-canto, garantizando que todas las grabaciones utilizadas provengan de fuentes públicas disponibles bajo licencias específicamente orientadas al uso científico. Esto asegura el respeto a los derechos de los colaboradores y contribuidores que aportan las grabaciones a la plataforma. Además, se realizó una revisión detallada de las condiciones de uso y las políticas de privacidad de Xeno-canto, garantizando el cumplimiento de estándares éticos y legales en la recolección y manejo de los datos. Este enfoque ético refuerza la credibilidad del proyecto y garantiza que su desarrollo sea sostenible y alineado con las buenas prácticas científicas y ambientales.

6. Resultados

Los objetivos del proyecto fueron alcanzados utilizando la metodología CRISP-ML(Q), la cual proporciona una estructura clara y organizada mediante la división del trabajo en tareas y actividades específicas para cada fase. Esta metodología facilitó la sistematización del proceso, asegurando un enfoque eficiente en el cumplimiento de los objetivos. A continuación, se presentan los resultados obtenidos en cada fase, comenzando con el **Objetivo 1**: Ajustar el modelo ResNet-50 con un conjunto de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris..

6.1. Objetivo 1: Ajustar el modelo ResNet-50 con un conjunto de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

6.1.1. Comprensión de datos y negocio

La revisión de literatura confirmó la relevancia de utilizar redes neuronales convolucionales (CNN) para clasificar especies a partir de grabaciones de audio, destacando su eficacia para identificar patrones acústicos específicos en entornos complejos. Además, se subrayó la importancia de los espectrogramas como representaciones espectrales clave para modelar las vocalizaciones, así como la necesidad de contar con datasets limpios y balanceados para mejorar la capacidad predictiva de los modelos. Asimismo, se resaltó el potencial de la transferencia de aprendizaje, especialmente en contextos con grabaciones limitadas, como una estrategia eficaz para maximizar la precisión del modelo (ver sección **Trabajos Relacionados**).

Además, la entrevista con el experto en machine learning, Ingeniero Oscar Cumbicus, aportó información clave sobre aspectos críticos y desafíos técnicos específicos de la clasificación de vocalizaciones, proporcionando recomendaciones fundamentales para optimizar el modelo. En la **Tabla 3** se resumen las respuestas más relevantes:

Tabla 3: Resumen de preguntas y respuestas de la entrevista

Pregunta	Respuesta
¿Cuáles son los principales desafíos en la clasificación de sonidos, considerando la complejidad de señales en entornos?	Uno de los mayores desafíos es el ruido ambiental que se mezcla con las vocalizaciones en entornos naturales, lo cual complica la identificación precisa de las especies, ya que las señales de los cantos pueden verse afectadas por esta variabilidad.

Pregunta	Respuesta
¿Cómo visualiza la aplicación del transfer learning para mejorar la precisión en la clasificación de sonidos?	El transfer learning es esencial para adaptar modelos a características específicas, mejorando tanto la precisión como la capacidad de generalización del modelo para clasificar sonidos en un rango amplio de señales.
¿Qué modelos preentrenados considera ideales para la clasificación de vocalizaciones de aves?	Recomendó modelos como VGG16, ResNet y Inception, preentrenados en conjuntos como ImageNet, los cuales pueden ajustarse para interpretar espectrogramas de vocalizaciones de aves.
¿Cuál sería su enfoque para ajustar los hiperparámetros durante el transfer learning?	Sugirió iniciar con una tasa de aprendizaje baja, ajustándola gradualmente según el rendimiento en validación. Monitorear este proceso permite optimizar el aprendizaje y la precisión del modelo.
¿Qué métricas considera adecuadas para evaluar el rendimiento del modelo?	Indicó el uso de métricas como precisión, recall y F1-score para casos de clases desbalanceadas, además de la matriz de confusión para evaluar verdaderos y falsos positivos en cada clase

Para acceder a la entrevista completa, se puede observar en la plataforma de YouTube [1] .

6.1.2. Ingeniería de datos (Preparación de datos)

Esta fase incluyó la construcción del dataset, la organización de los datos y su etiquetado, así como la aplicación de criterios específicos para mantener la integridad y representatividad de las grabaciones.

6.1.2.1. Tarea 1: Selección de datos.

Para garantizar la calidad y relevancia de los datos, se definieron criterios específicos que permitieron seleccionar grabaciones adecuadas para el análisis de las vocalizaciones de las especies en estudio. En la **Tabla 4** se describen los criterios aplicados para la selección de datos.

[1] Entrevista YouTube

Tabla 4: Criterios de selección de grabaciones de audio

Criterio	Descripción
Calidad del audio	Mínimo 16 bits. Tasa de muestreo de 8KHz hasta 48KHz. Sin distorsiones audibles o ruidos de fondo significativos.
Duración mínima	Mínimo 5 segundos para asegurar una muestra representativa de la vocalización.
Contexto ambiental	Grabaciones realizadas en diversos entornos (zonas urbanas y rurales), aunque el ruido ambiental no debe ser predominante.
Vocalización representativa	Debe incluir vocalizaciones características de cualquiera de las tres especies seleccionadas (Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris). Preferentemente vocalizaciones durante su periodo de actividad habitual (amanecer o atardecer).

Tras definir los criterios, se seleccionaron y descargaron grabaciones desde la plataforma Xeno-canto[2], específicamente para las especies Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

Para optimizar este proceso, se empleó web scraping, lo que permitió automatizar la extracción de grabaciones específicas por especie, basándose en sus nombres comunes en inglés. En la **Figura 10** se presenta el código utilizado para automatizar la extracción y organización de las grabaciones desde la plataforma Xeno-canto.

```
# Nombres comunes en inglés de las especies a buscar
common_names = ['Blackish Tapaculo', 'Plain-tailed Wren', 'Grey-breasted Wood Wren']

# Crear un DataFrame vacío para almacenar los resultados
result_df = pd.DataFrame()

# Función para realizar la solicitud al API de Xeno-Canto por nombre común
def fetch_species_data(common_name):
    response = requests.get(f'https://www.xeno-canto.org/api/2/recordings?query="{common_name}"')
    return response.json()

# Bucle para obtener datos de cada nombre común
for name in common_names:
    species_data = fetch_species_data(name)
    ids, files, file_names, ens, lengths, gens = [], [], [], [], [], []
    for recording in species_data["recordings"]:
        ids.append(recording["id"])
        files.append(recording["file"])
        file_names.append(recording["file-name"])
        ens.append(recording["en"])  # Nombre común
        lengths.append(recording["length"])
        gens.append(recording["gen"])  # Género

    # Guardar datos en DataFrame
    df_ = pd.DataFrame({'id': ids, "file": files, "file-name": file_names, "en": ens, "gen": gens, "length": lengths})
    result_df = pd.concat([result_df, df_], ignore_index=True)
```

Figura 10: Web scraping técnica implementada para la descarga de grabaciones de audio de las tres especies de aves

[2] Plataforma Xeno-canto

Inicialmente, se recopilaron 1,185 grabaciones de audio correspondientes a las vocalizaciones de las tres especies estudiadas. Estas grabaciones se almacenaron en un repositorio público de Google Drive[3], garantizando su trazabilidad y accesibilidad a lo largo del proyecto. Además, se generó el archivo CSV denominado species_recordings.csv[4], que contiene la información estructurada de cada grabación, En la **Tabla 5** se presentan los parámetros y su descripción.

Tabla 5: Parámetros en la descarga de grabaciones

Parámetro	Descripción
id	Número de referencia único asignado a cada grabación para su identificación.
file	Ruta de descarga del archivo, indicando la ubicación exacta de cada grabación en el repositorio de Google Drive.
file-name	Nombre del archivo de audio, utilizado para diferenciar cada grabación.
en	Nombre común de cada especie en inglés, lo que permite identificar la especie vocalizadora.
gen	Género taxonómico de la especie, clasificando cada ave dentro de su respectiva familia.
length	Duración de la grabación en segundos, útil para el análisis de la longitud de las vocalizaciones.

En la **Figura 11** se muestra un fragmento de código que crea un archivo CSV denominado audios.csv[5] a partir de species_recordings.csv. El código normaliza los nombres de archivo a minúsculas, genera una nueva columna que concatena el 'id' con la extensión del archivo y filtra las grabaciones descargadas. Además, reemplaza los nombres de las especies en inglés por sus equivalentes en español: "Blackish Tapaculo" se convierte en "Churrín Negruzco", "Plain-tailed Wren" en "Soterrey Cola Pálida", y "Grey-breasted Wood Wren" en "Soterrey Montés de Pecho Gris". Este proceso asegura que el modelo de clasificación utilice datos adecuados para las especies seleccionadas.

[3] Repositorio Google Drive
[4] CSV species_recordings
[5] CSV audios

```python
# Asegurarse de que los nombres de archivo estén en minúsculas
df['file-name'] = df['file-name'].str.lower()

# Concatenar 'id' y la extensión del archivo
df['filename'] = df['id'].astype(str) + df['file-name'].str[-4:]  # Usar los últimos 4 caracteres como extensión

# Filtrar los archivos que se han descargado
audios = df[df['filename'].isin(files)][['filename', 'en']]

# Reemplazar los nombres en inglés por los nombres en español
name_replacements = {
    "Blackish Tapaculo": "Churrín Negruzco",
    "Plain-tailed Wren": "Soterrey Cola Pálida",
    "Grey-breasted Wood Wren": "Soterrey Montés de Pecho Gris"
}

audios['en'] = audios['en'].replace(name_replacements)

# Guardar los resultados en un nuevo archivo CSV
audios.to_csv(os.path.join(csv_dir, 'audios.csv'), index=False)
```

Figura 11: Parámetros necesarios con nombres de especies en español

En la **Figura 12** se presenta un nuevo fragmento de código que crea un archivo CSV denominado audios_modified.csv[6]. En este proceso, la columna 'en' del archivo original ha sido renombrada a 'category'. Este cambio es fundamental para alinear los datos con la estructura requerida para el modelo de clasificación, facilitando la identificación de las especies de aves correspondientes a cada grabación.

```python
# Renombrar la columna 'en' a 'category'
df.rename(columns={'en': 'category'}, inplace=True)
```

Figura 12: Columna 'en' renombrada a 'category'

6.1.2.2. Tarea 2: Ingeniería de características.

La ingeniería de características se enfocó en la transformación y creación de variables para mejorar la capacidad del modelo de aprender y hacer predicciones precisas. En la **Figura 13** se muestra el código utilizado para procesar y convertir archivos de audio de formato WAV a MP3 con la biblioteca pydub, además de verificar la validez de los archivos convertidos. El código leyó el archivo audios_modified.csv, que contiene los nombres de los archivos y sus categorías, y creó un directorio para almacenar los archivos MP3 convertidos[7]. La función de validación identificó errores al abrir los archivos MP3, clasificándolos en listas de válidos e inválidos. Finalmente, se generaron dos archivos CSV: uno para los archivos válidos[8] (incluyendo los ya existentes y los convertidos) y otro para los inválidos[9], ambos con la columna 'category' que indica la especie correspondiente, asegurando que los datos de audio estén organizados y listos para el modelado.

[6] CSV audios_modified
[7] Almacenamiento archivos mp3
[8] CSV archivos válidos
[9] CSV archivos inválidos

```python
# Recorrer los nombres de los archivos en el DataFrame
for fn in tqdm(df['filename']):
    input_path = os.path.join(audio_dir, fn)
    output_path = os.path.join(mp3_dir, fn[:-4] + '.mp3')  # Cambiar la extensión a MP3

    if fn.endswith('.wav'):  # Convertir solo archivos WAV a MP3
        try:
            sound = AudioSegment.from_wav(input_path)
            sound.export(output_path, format='mp3')  # Convertir a MP3
            valid_files.append(fn[:-4] + '.mp3')  # Añadir el nombre de archivo convertido a la lista de válidos
        except Exception as e:
            print(f"Error converting {input_path} to MP3: {e}")
            invalid_files.append(fn)  # Añadir archivos inválidos si hay error

    elif fn.endswith('.mp3'):  # Verificar archivos MP3 ya existentes
        if is_valid_mp3(input_path):
            valid_files.append(fn)  # Solo agregar archivos válidos
        else:
            print(f"Invalid MP3 file: {input_path}")
            invalid_files.append(fn)  # Añadir archivos inválidos
```

Figura 13: Conversión de WAV a MP3 y archivos inválidos

Se creó una carpeta específica para almacenar los archivos MP3 identificados como inválidos[10] durante el procesamiento de las grabaciones. Estos archivos fueron clasificados como problemáticos por diversas razones, lo que impidió su conversión o procesamiento adecuado en la fase de ingeniería de características.

En la **Tabla 6** se detallan las principales razones de invalidez de 21 archivos MP3, explicando los factores que llevaron a su exclusión.

Tabla 6: Razones de invalidez de archivos

Razón de invalidez	Descripción
Descarga incompleta	Algunos archivos MP3 pueden no haberse descargado completamente debido a interrupciones en la conexión o problemas con el servidor de Xeno-canto, resultando en archivos incompletos o dañados.
Codificación incorrecta	Aunque el formato MP3 es estándar, ciertas grabaciones pueden haber sido codificadas incorrectamente o con ajustes no compatibles con bibliotecas de procesamiento de audio como pydub, impidiendo su correcto procesamiento.
Grabaciones defectuosas	Las grabaciones en Xeno-canto son aportadas por usuarios de distintas regiones, por lo que algunos archivos podrían haber sido mal grabados o editados, resultando en archivos defectuosos.
Metadatos dañados	Los archivos MP3 pueden contener metadatos adicionales (como nombre del artista o título de la pista); si estos están dañados, pueden impedir que herramientas de procesamiento de audio abran o lean correctamente el archivo.

[10] Almacenamiento archivos mp3 inválidos

De los 1,185 archivos recopilados inicialmente, se excluyeron 21 debido a su invalidez, dejando un total de 1,164 grabaciones válidas para el análisis. Estas grabaciones se distribuyen de la siguiente manera: 500 corresponden a Soterrey Montés de Pecho Gris, 451 a Churrín Negruzco y 213 a Soterrey Cola Pálida.

En la **Figura 14** se muestra esta distribución en un diagrama de barras, donde se observa la variabilidad en el número de grabaciones entre las especies, destacando al Soterrey Montés de Pecho Gris como la más representada.

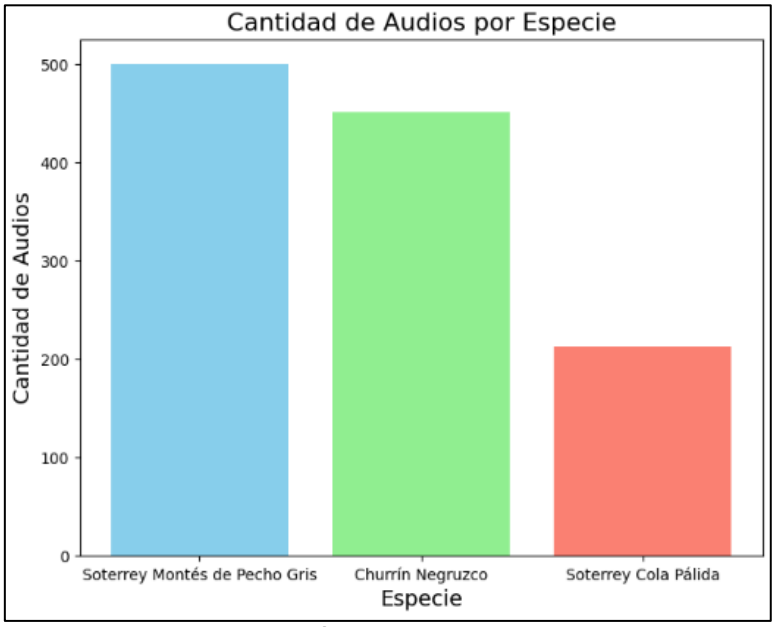


Figura 14: Distribución original de audios por especie

Para continuar con la preparación de los datos, fue esencial identificar y estandarizar las tasas de muestreo de los 1,164 archivos de audio válidos. Determinar la tasa de muestreo de cada archivo aseguró la consistencia en el procesamiento y análisis, ya que las variaciones pueden impactar la calidad y precisión de los modelos de clasificación. En la **Figura 15** se muestra el código utilizado para obtener la tasa de muestreo de cada archivo, lo que representó un paso clave en la estandarización de los datos.

```
# Lista para almacenar las tasas de muestreo de cada archivo
sample_rates = []

# Procesar cada archivo MP3 y obtener la tasa de muestreo
for file in tqdm(files):
    try:
        # Cargar el archivo MP3
        file_path = os.path.join(mp3_dir, file)
        audio = AudioSegment.from_mp3(file_path)

        # Obtener la tasa de muestreo
        sample_rate = audio.frame_rate
        sample_rates.append({'filename': file, 'sample_rate': sample_rate})
    except Exception as e:
        print(f"Error al procesar {file}: {e}")
        sample_rates.append({'filename': file, 'sample_rate': 'error'})
```

Figura 15: Obtención de la tasa de muestreo de los 1,164 audios

Las tasas de muestreo obtenidas se almacenaron en un nuevo archivo CSV titulado sample_rates_mp3_files.csv[11], que contiene dos columnas: filename, con el nombre de cada archivo en formato .mp3, y sample_rate, que indica la tasa de muestreo correspondiente a cada grabación. En la **Figura 16** se muestra el código utilizado para contar las tasas de muestreo, lo que permite visualizar su distribución y tomar decisiones sobre su estandarización.

```
# Agrupar por la columna 'sample_rate' y contar la cantidad de archivos en cada grupo
sample_rate_counts = df_sample_rates['sample_rate'].value_counts()

# Mostrar el resultado
print(sample_rate_counts)
```

```
sample_rate
44100    932
48000    208
32000     12
22050      9
24000      2
16000      1
Name: count, dtype: int64
```

Figura 16: Conteo de tasas de muestreo por cada audio

La distribución de las tasas de muestreo en el conjunto de datos muestra que la mayoría de los archivos de audio, un total de 932, están en 44,100 Hz, siendo esta la frecuencia predominante. Para asegurar la consistencia en el procesamiento y evitar problemas de compatibilidad en el modelo, todas las grabaciones se convertirán o "resamplearán" a esta tasa estándar. Este proceso garantizará un análisis uniforme y que los audios mantengan la misma calidad y formato. En la **Figura 17** se muestra el código utilizado para realizar el resampleo de los archivos que no están en 44,100 Hz, enfocándose únicamente en aquellos con tasas diferentes.

```
# Cargar el archivo MP3
audio = AudioSegment.from_mp3(os.path.join(mp3_dir, file))

# Verificar la tasa de muestreo (si no es 44100Hz, resamplear)
if audio.frame_rate != 44100:
    print(f"Resampleando {file} de {audio.frame_rate} Hz a 44100 Hz.")
    audio = audio.set_frame_rate(44100)

# Guardar el archivo resampleado
output_path = os.path.join(resample_dir, file)
audio.export(output_path, format='mp3')
```

Figura 17: Resampleo de audios a 44,100Hz

También, se generó un archivo CSV denominado resampled_mp3_files.csv[12], que conserva la estructura original con dos columnas: filename, que corresponde al nombre de cada archivo de audio, y category, que identifica la especie asociada a cada grabación. En este archivo, todos los audios están ahora en la tasa de muestreo estándar de 44,100 Hz,

[11] CSV tasas de muestreo
[12] CSV archivos mp3 resampleados

asegurando la uniformidad requerida para las siguientes etapas del análisis y procesamiento de datos.

6.1.2.3. Tarea 3: Estandarización de datos.

Aumento y balanceo de datos

Para aplicar la técnica de aumento de datos y balancear las clases minoritarias, se creó la carpeta denominada complete_dataset[13], donde se trasladaron los 1,164 archivos preprocesados. Esta carpeta facilitó el aumento de datos para las clases con menos de 500 muestras, logrando una representación equitativa entre las especies. En la **Figura 18** se muestra el código utilizado para estandarizar la cantidad de audios a 500 por especie, aplicando transformaciones como el cambio de tono y la adición de ruido controlado para simular un ambiente natural. Cabe destacar que la clase Soterrey Montés de Pecho Gris ya contaba con las 500 muestras requeridas, por lo que no fue necesario aplicar aumento de datos en esta categoría.

```
# Función para aplicar transformaciones a un audio
def augment_audio(audio, sr):
    # Convertir a mono si tiene más de un canal
    if len(audio.shape) > 1:
        audio = librosa.to_mono(audio)
    # Cambiar el tono
    y_pitch = librosa.effects.pitch_shift(audio, sr=sr, n_steps=4)  # Subir 4 semitonos
    # Añadir ruido
    noise = np.random.randn(len(audio))
    y_noise = audio + 0.005 * noise
    return [y_pitch, y_noise]
# Cargar el audio original
y, sr = librosa.load(audio_file_path, sr=None)

# Aplicar las transformaciones de aumento de datos
augmented_audios = augment_audio(y, sr)

# Guardar los archivos aumentados
for j, aug_audio in enumerate(augmented_audios):
    if len(new_rows) + current_count >= target_count:  # Detener si alcanzamos los 500
        break

    # Crear el nombre de archivo aumentado
    augmented_filename = f"{complete_dataset_dir}{row['filename'].replace('.mp3', f'_aug{j}.mp3')}"
    sf.write(augmented_filename, aug_audio, sr)  # Guardar el audio aumentado
    print(f"Guardado: {augmented_filename}")  # Confirmar guardado
```

Figura 18: Aumento de datos a las clases minoritarias

El proceso de aumento de datos se documentó en un archivo CSV denominado complete_balanced_dataset.csv[14]. En este archivo, los nuevos audios generados se nombraron con un sufijo indicativo de la transformación aplicada, como **_aug0** y **_aug1**, permitiendo hasta dos versiones adicionales por archivo y facilitando el balanceo deseado. Sin embargo, tras el aumento inicial, la clase Soterrey Cola Pálida alcanzó solo 451 audios, quedando 49 por debajo del equilibrio con las demás especies. Esto se debió a que las

[13] Dataset completo
[14] CSV dataset completo balanceado

transformaciones aplicadas no generaron suficiente variabilidad en los audios originales, dada la cantidad inicial limitada a 213 grabaciones.

Para resolverlo, se utilizó un código adicional que generó los 49 audios faltantes, completando el balance de 500 muestras por especie. En la **Figura 19** se presenta el código empleado para esta corrección final.

```
# Aplicar aumentaciones
augmented_audios = augment_audio(audio_samples, sr)

# Guardar los nuevos audios aumentados
for j, aug_audio in enumerate(augmented_audios):
    if len(new_rows) >= num_to_augment:
        break

    # Crear un nuevo nombre para los archivos aumentados
    augment_filename = f"{os.path.splitext(row['filename'])[0]}_aug{j}.mp3"
    augment_filepath = os.path.join(augmented_dir, augment_filename)

    # Guardar el archivo aumentado en MP3 usando pydub
    augmented_segment = AudioSegment(
        (aug_audio * 32767).astype(np.int16).tobytes(),  # Convertir de float a int16
        frame_rate=sr,
        sample_width=audio.sample_width,
        channels=audio.channels
    )
    augmented_segment.export(augment_filepath, format="mp3")

    # Añadir la información del nuevo archivo al CSV
    new_rows.append({
        'filename': augment_filename,
        'category': 'Soterrey Cola Pálida'
    })
```

Figura 19: Aumento de 49 archivos de audio para la clase Soterrey Cola Pálida

Esta actualización se documentó en un nuevo archivo CSV denominado updated_complete_balanced_dataset.csv[15], que incluye un total de 1,500 archivos de audio correctamente etiquetados por especie, incorporando los audios generados mediante el aumento. En la **Figura 20** se presenta un diagrama de barras que ilustra la cantidad de audios por especie tras el aumento y balanceo, destacando el éxito de la estrategia implementada en la estandarización de los datos.

[15] CSV dataset completo balanceado actualizado

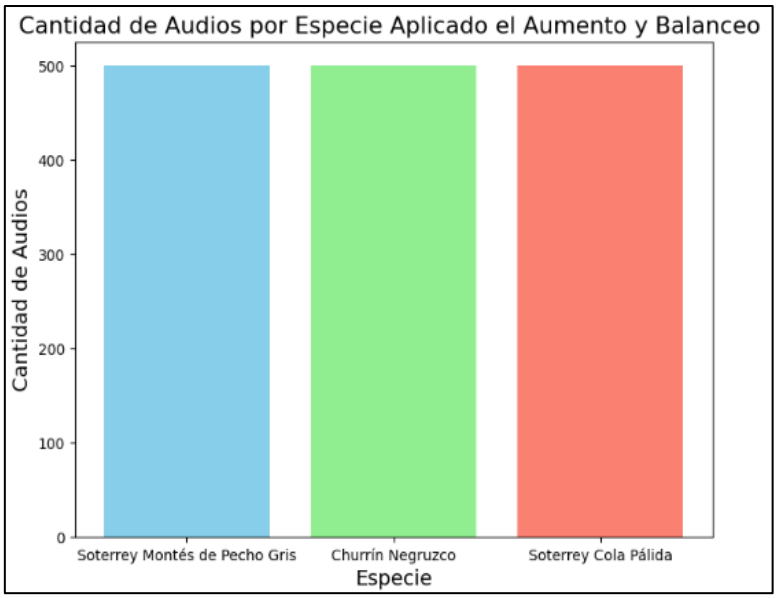


Figura 20: Distribución de audios por especie ya aplicado el aumento y balanceo

Estratificación y División del Dataset

La estratificación aseguró una representación equitativa de las clases en los conjuntos de entrenamiento y prueba. En este proyecto, el dataset completo se dividió estratégicamente, asignando el 90% de las muestras totales de cada especie (1,350 muestras) para el entrenamiento y el 10% restante (150 muestras) para las pruebas.

A diferencia de otras estrategias que duplican archivos, se optó por reutilizar las grabaciones existentes, distribuyendo algunas en la carpeta **train**[16] y otras en la carpeta **test**[17], optimizando así el uso del almacenamiento disponible en Google Drive. En la **Figura 21** se presenta el código implementado para realizar esta estratificación de manera eficiente.

```
# Definir la cantidad de archivos para train y test por especie
train_count = 450
test_count = 50

# Agrupar los datos por especie
grouped = df.groupby('category')

for name, group in grouped:
    print(f"Procesando especie: {name}")

    # Asegurarse de que haya suficientes audios
    if len(group) >= (train_count + test_count):
        # Mezclar aleatoriamente los archivos
        shuffled_group = group.sample(frac=1).reset_index(drop=True)

        # Obtener los archivos para train y test
        train_files = shuffled_group.iloc[:train_count]
        test_files = shuffled_group.iloc[train_count:train_count + test_count]
```

Figura 21: Estratificación para train y test

[16] Conjunto de datos train
[17] Conjunto de datos test

En la **Figura 22** se presenta un diagrama circular que ilustra el porcentaje y la cantidad de muestras asignadas a los conjuntos de entrenamiento y prueba.

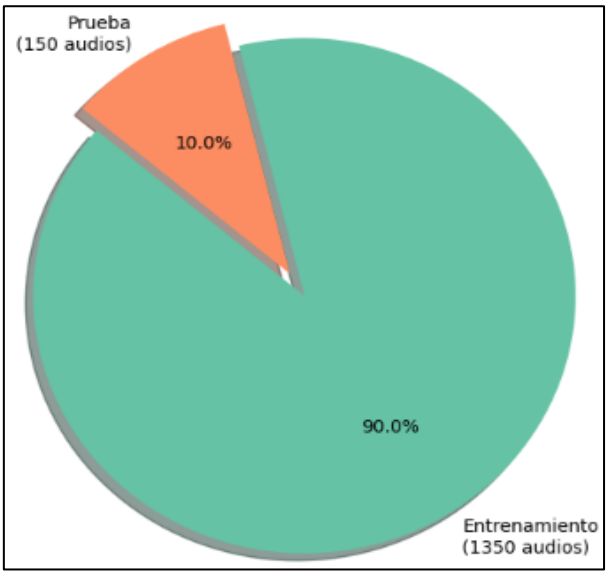


Figura 22: Distribución de archivos en conjuntos de train y test

Preparación de bloques de datos (DataBlock) y cargadores de datos (DataLoader)

Antes de crear el DataBlock, se definieron los dataframes **df_train**[18] y **df_test**[19] para representar la estratificación realizada previamente en los conjuntos de entrenamiento y prueba. Esto garantiza que los datos estén bien estructurados y sean fácilmente accesibles para el Datablock. En la **Figura 23** se muestra el código utilizado para definir estos dataframes, asegurando la proporción adecuada de cada clase en ambos conjuntos.

```
# Filtrar los DataFrames según los archivos de entrenamiento y prueba
df_train = all_species[all_species['filename'].isin(train_files)]
df_test = all_species[all_species['filename'].isin(test_files)]

# Mostrar las dimensiones de los DataFrames de entrenamiento y prueba
print(f"Dimensiones del DataFrame de entrenamiento: {df_train.shape}")
print(f"Dimensiones del DataFrame de prueba: {df_test.shape}")
```

```
Dimensiones originales del DataFrame: (1500, 2)
Dimensiones del DataFrame de entrenamiento: (1350, 2)
Dimensiones del DataFrame de prueba: (150, 2)
```

Figura 23: Definición de dataframes para train y test

Para mayor claridad y transparencia, en la **Figura 24** se presenta el conteo de las especies en el dataframe **df_train**, que contiene 450 audios para especie: Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris. Esta distribución equitativa asegura una representación adecuada de cada clase en el conjunto de entrenamiento, favoreciendo un aprendizaje balanceado del modelo.

[18] CSV df_train
[19] CSV df_test

```
# Contar cuántos audios originales hay por especie
print(df_train['category'].value_counts())

category
Churrín Negruzco                450
Soterrey Cola Pálida            450
Soterrey Montés de Pecho Gris   450
Name: count, dtype: int64
```

Figura 24: Verificación de datos en df_train

En la **Figura 25** se presenta el conteo de las especies en el dataframe **df_test**, que incluye 50 audios por cada especie: Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris. Esta organización mantiene el equilibrio entre las clases en el conjunto de prueba, permitiendo una evaluación justa y precisa del modelo con una muestra representativa de cada categoría.

```
# Verificar cuántas entradas hay por categoría
print(df_test['category'].value_counts())

category
Churrín Negruzco                50
Soterrey Cola Pálida            50
Soterrey Montés de Pecho Gris   50
Name: count, dtype: int64
```

Figura 25: Verificación de datos en df_test

Por otro lado, se aplicaron transformaciones con las herramientas **FastAI** y **Fastaudio** para optimizar los datos de audio durante el entrenamiento del modelo. Estas transformaciones, a diferencia del aumento y balanceo realizados previamente, se ejecutan en tiempo real, permitiendo un preprocesamiento dinámico sin incrementar el tamaño del dataset.

Las transformaciones pueden aplicarse a nivel de **ítem**, ajustando individualmente cada archivo de audio para garantizar la calidad y consistencia, o a nivel de **batch**, procesando grupos de archivos simultáneamente para mejorar la variabilidad y robustez del modelo. En la **Figura 26** se muestran las transformaciones aplicadas a nivel de ítem y batch.

```
# Configuración del espectrograma
seconds = 15
cfg = AudioConfig.BasicMelSpectrogram(n_fft=1024)
a2s = AudioToSpec.from_cfg(cfg)

# Transformaciones de items
item_transforms = [
    ResizeSignal(seconds * 1000, pad_mode=AudioPadType.Repeat), # Redimensionar audio
    AddNoise(0.005),                # Añadir ruido
    Normalize(),                    # Normalizar audio
    a2s                             # Convertir a espectrograma
]

# Transformaciones de batch
batch_transforms = [
    ChangeVolume(),                 # Cambiar volumen
]
```

Figura 26: Transformaciones por ítem y batch

Transformaciones por ítem:

- **ResizeSignal**: Ajusta la duración de los audios a un intervalo fijo de 15 segundos. Utilizando el modo de relleno Repeat (repetir), los audios más cortos se repiten hasta alcanzar la duración deseada, mientras que los audios más largos se recortan, garantizando una longitud uniforme para cada espectrograma generado. Esto permite que el modelo procese las entradas de manera eficiente y mantiene el tamaño del dataset manejable, sin generar archivos adicionales que ocupen espacio de almacenamiento innecesario.
- **AddNoise**: Añade un leve ruido controlado al audio con un nivel de **0.005**. Esta técnica simula condiciones del mundo real, donde las grabaciones pueden verse afectadas por diferentes tipos de interferencias. Al introducir ruido en el audio, se entrena al modelo para que sea más robusto y capaz de generalizar mejor ante variaciones en el sonido de fondo, mejorando así su desempeño en entornos no controlados.
- **Normalize**: Normaliza la señal de audio ajustando sus valores a una escala uniforme. Este proceso es fundamental para reducir las variaciones de amplitud entre los archivos, asegurando que los niveles de volumen sean consistentes. Esto facilita la comparación entre los audios y evita que el modelo se sesgue hacia grabaciones más fuertes o débiles, lo que podría afectar negativamente su capacidad de aprendizaje.
- **AudioToSpec**: Convierte el audio en un espectrograma mel utilizando la configuración **AudioConfig.BasicMelSpectrogram**, con una cantidad de puntos (**n_fft**) de 1024. Esta transformación proporciona una representación visual de las frecuencias a lo largo del tiempo, lo que permite al modelo detectar patrones y características de los sonidos de manera más efectiva. Es importante señalar que los espectrogramas no se guardan como imágenes, sino que se procesan internamente en el modelo durante el entrenamiento. Al trabajar con espectrogramas de esta manera, el modelo puede aprender a identificar las vocalizaciones de las especies de aves con mayor precisión.

Transformaciones a nivel de lote o batch:

- **ChangeVolume**: Ajusta el volumen del audio de manera aleatoria en cada lote. Esta variación permite que el modelo se adapte a grabaciones con diferentes niveles de volumen, lo que es común en los datos del mundo real. Al incluir esta transformación, se fortalece la capacidad de generalización del modelo, preparándolo mejor para manejar variaciones en las grabaciones durante el proceso de inferencia.

Estas transformaciones mejoran la consistencia y calidad de los datos de entrada, facilitando un flujo continuo y homogéneo durante el entrenamiento. Al realizarse en tiempo real, optimizan el manejo de datos y garantizan un preprocesamiento adecuado sin incrementar el uso de almacenamiento.

Creación de DataBlock

En este caso, el DataBlock fue utilizado para organizar, preprocesar y estructurar los datos de manera eficiente antes del entrenamiento del modelo. Se configuró con las siguientes especificaciones, permitiendo integrar parámetros clave como las transformaciones aplicadas a los audios, las clases de especies y la proporción entre los conjuntos de entrenamiento y validación.

- **Definición de bloques**: Se especifican dos bloques principales, uno para el audio y otro para la categoría. Para el bloque de audio, se emplea **AudioBlock** con la configuración **crop_signal_to**, que asegura que cada señal de audio se recorte a la duración definida (15 segundos en milisegundos), mientras que el bloque de categoría CategoryBlock se utiliza para etiquetar cada archivo con la especie de ave correspondiente.
- **Extracción de archivos de audio**: Mediante **get_x=ColReader("filename", pref=Path(train_dir))**, se indica que los archivos de audio deben obtenerse desde la columna "filename" en el archivo CSV, y se establece una ruta de acceso base (train_dir) donde se encuentran los archivos.
- **División del conjunto de datos**: Utilizando **splitter=TrainTestSplitter**, los datos se dividen en un 80% para entrenamiento y un 20% para validación. Este método realiza una división estratificada, garantizando una representación equitativa de cada categoría (especie de ave) en ambos subconjuntos. Se incluye una semilla (**random_state=42**) para asegurar la reproducibilidad de los resultados. En el contexto de machine learning, este valor se utiliza para fijar el estado inicial del generador de números aleatorios, permitiendo que los mismos datos se asignen a los conjuntos de entrenamiento y validación en cada ejecución del código. El valor específico **42** es comúnmente utilizado en ejemplos y prácticas de machine learning debido a su popularidad en la comunidad (es una referencia cultural, además de ser un número en la media de rangos seguros para semillas). Este valor logra un equilibrio, siendo lo suficientemente grande como para evitar problemas de colisión en secuencias de aleatoriedad y lo suficientemente pequeño como para evitar problemas de overflow en ciertas implementaciones numéricas. Escoger valores mucho mayores no aporta mejoras significativas en reproducibilidad y podría incrementar el tiempo de procesamiento en algunos casos.
- **Transformaciones aplicadas:** Incluyen ajustes de duración, introducción de ruido controlado para variabilidad, normalización de amplitud, y conversión a espectrogramas, como ya se detalló anteriormente. Estas transformaciones, tanto a nivel de ítem como de lote, optimizan el entrenamiento al ofrecer variabilidad dentro de límites controlados y mejorar la robustez del modelo frente a variaciones comunes en grabaciones de campo.

En la **Figura 27** se muestra el código empleado para crear el DataBlock, integrando todos estos parámetros y transformaciones.

```
# Crear el DataBlock
train_dir = resample_dir + 'train'
auds = DataBlock(
    # Definir los bloques: Audio y Categoría
    blocks=(AudioBlock(crop_signal_to=seconds * 1000), CategoryBlock),
    # Obtener los audios desde la columna "filename" del CSV
    get_x=ColReader("filename", pref=Path(train_dir)),
    # División de los datos: 80% para entrenamiento, 20% para validación
    splitter=TrainTestSplitter(random_state=42, stratify=df_train['category'], test_size=0.2),
    # Transformaciones a nivel de item (preparación del audio)
    item_tfms=item_transforms,
    # Transformaciones a nivel de batch
    batch_tfms=batch_transforms,
    # Obtener la categoría (especie de ave) desde la columna "category"
    get_y=ColReader("category")
)
```

Figura 27: Creación de DataBlock

Creación de DataLoader

El DataLoader fue utilizado para cargar los datos de manera eficiente durante el entrenamiento del modelo, dividiendolos en lotes manejables (batches) y aplicando las transformaciones necesarias en tiempo real. Esto permitió optimizar el flujo de datos desde el DataBlock hacia el modelo, asegurando un procesamiento consistente y fluido.

Para crear el DataLoader, se emplea el método **dataloaders** sobre el DataBlock configurado previamente, con el dataframe df_train y un tamaño de lote (batch_size) de 32. A continuación, se explica la justificación de estos elementos:

- **Uso de df_train**: En este caso, df_train contiene los datos de entrenamiento ya preprocesados y balanceados para asegurar una cantidad equitativa de muestras entre las especies de aves. Este dataframe es el que se emplea para cargar las instancias en el DataLoader y garantiza que la estructura y formato de los datos cumplen con las configuraciones del DataBlock. Esto permite que los datos se distribuyan correctamente en los lotes, manteniendo el balance entre clases y aprovechando la división previa de entrenamiento y validación.
- **Tamaño de lote (bs=32)**: El parámetro **bs** establece la cantidad de ejemplos que se procesarán simultáneamente en cada paso de entrenamiento o validación. Un tamaño de lote de 32 se considera óptimo en este caso porque permite un buen balance entre el uso de memoria y la estabilidad de los gradientes durante el entrenamiento. Este tamaño también ayuda a optimizar los tiempos de entrenamiento al reducir la frecuencia con la que se debe actualizar el modelo, sin comprometer la capacidad de aprendizaje de la red, lo cual es esencial cuando se entrena en Google Colab o en otros entornos con recursos de almacenamiento limitados.

En la **Figura 28** se presenta la línea de código que carga los datos en los DataLoaders, optimizando su manejo durante el entrenamiento y la validación del modelo.

```
# Cargar los datos en DataLoaders
dls = auds.dataloaders(df_train, bs=32)
```

Figura 28: Creación de DataLoader

Una vez creado el DataLoader, la herramienta FastAI transforma la forma de onda del audio en un espectrograma medinte la función AudioToSpec. Esta conversión utiliza configuraciones como n_fft, hop_length, y sample_rate, previamente definidas en el DataBlock, para representar visualmente las frecuencias a lo largo del tiempo. Esta transformación es clave para que el modelo detecte patrones y características específicas de las vocalizaciones de aves. El espectrograma no se guarda como una imagen, sino como una estructura de datos dentro de FastAI. Al convertir la forma de onda (representación temporal) en un espectrograma (representación tiempo-frecuencia), el modelo puede aprender de manera más efectiva a identificar patrones auditivos complejos que distinguen a las especies.

En la **Figura 29** se muestra la forma de onda del audio junto con su espectrograma de Mel correspondiente. En la forma de onda, el eje X representa el **tiempo** y el eje Y la **amplitud** de la señal. En el espectrograma de Mel, el eje X abarca hasta 15,000 ms, reflejando la duración del audio, mientras que el eje Y representa las frecuencias hasta 20000 Hz, resaltando las características sonoras que ocurren para distinguir las vocalizaciones entre especies de aves.

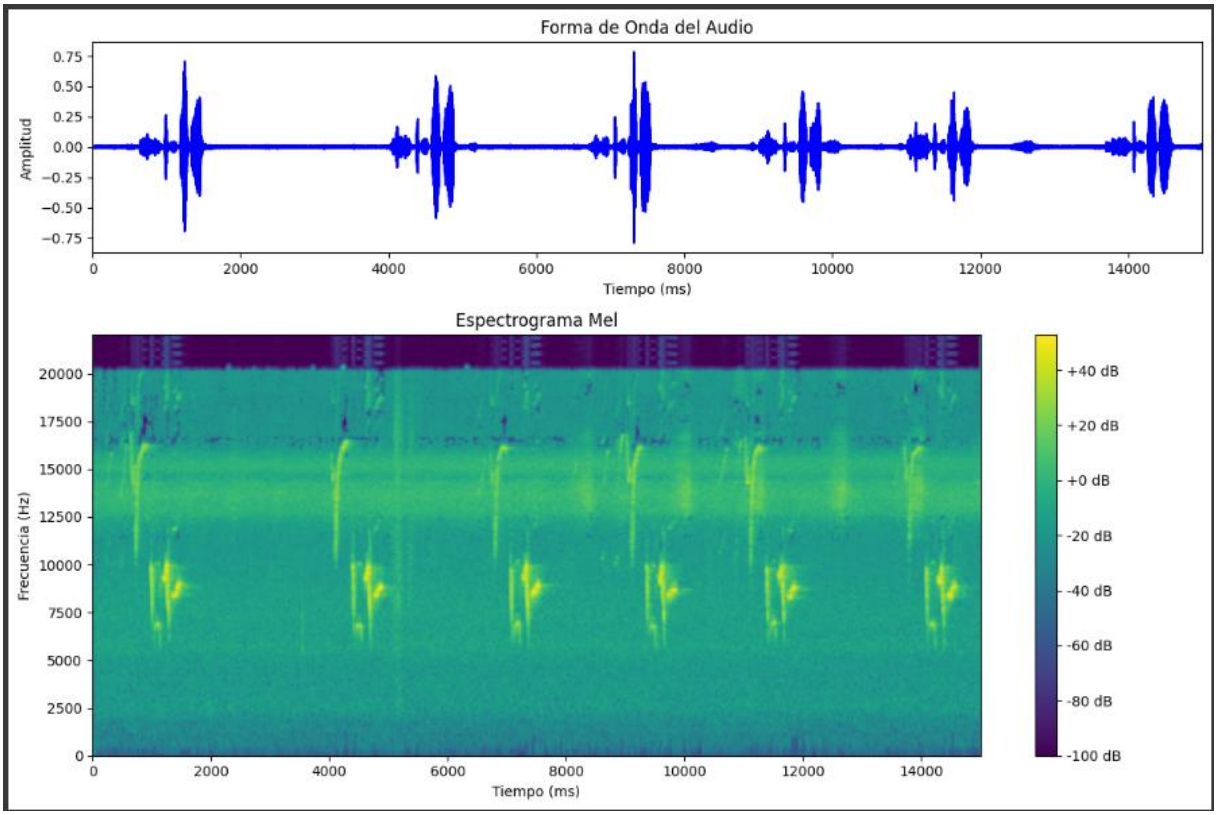


Figura 29: Transformación de forma de onda a espectrograma

En la **Figura 30** se presentan tres espectrogramas correspondientes a muestras aleatorias de las especies estudiadas. Cada espectrograma destaca las características acústicas únicas de cada vocalización, permitiendo visualizar cómo las diferentes frecuencias se distribuyen a lo largo del tiempo.

- **Soterrey Montés de Pecho Gris**: El espectrograma correspondiente a esta especie destaca su singularidad acústica, permitiendo diferenciarlo de las otras especies presentadas.
- **Churrín Negruzco**: El espectrograma de esta especie muestra patrones distintivos en su canto, con frecuencias predominantes que pueden ser útiles para su identificación.
  Soterrey Cola Pálida: En este espectrograma, se pueden observar variaciones en la amplitud y frecuencia, reflejando el estilo particular de vocalización de esta ave.

Cada espectrograma está diseñado para facilitar la identificación de las vocalizaciones de estas aves, destacando las características sonoras clave para su clasificación.

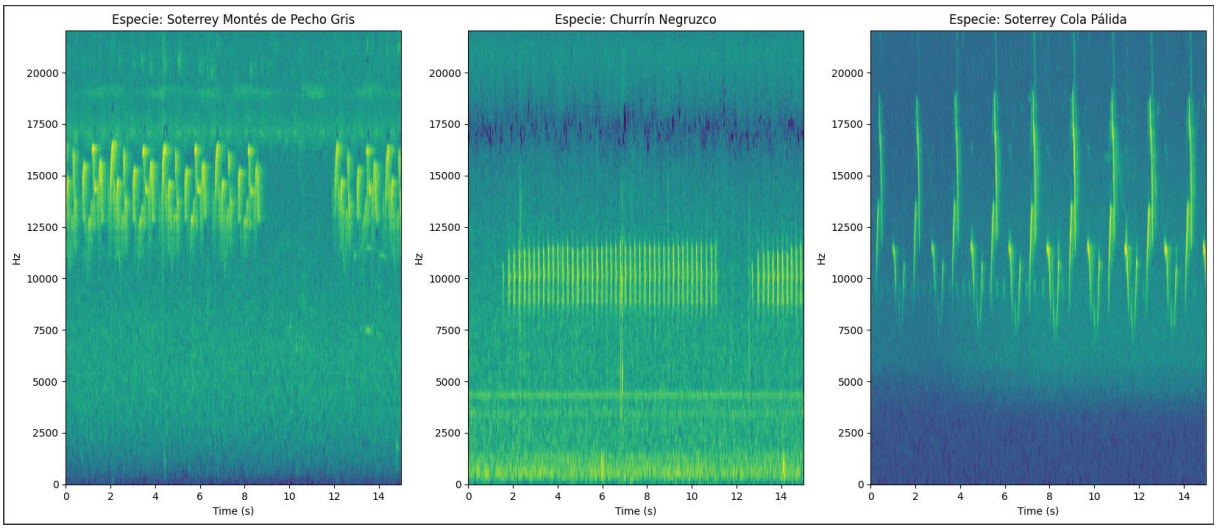


Figura 30: Lote de espectrogramas

6.1.3. Ingeniería de modelos de aprendizaje automático

6.1.3.1. Tarea 1: Selección de los hiperparámetros.

En esta tarea, se realizó un enfoque sistemático para seleccionar y ajustar los hiperparámetros, esenciales para el éxito del modelo ResNet-50 en la clasificación de vocalizaciones de aves. El proceso comenzó adaptando la arquitectura de la red con una capa de entrada configurada para recibir señales de audio, estableciendo el parámetro **n_in=1** para un solo canal de entrada.

Para evaluar el rendimiento del modelo, se utilizó la función de pérdida **CrossEntropyLossFlat**, adecuada para tareas de clasificación multi-clase, junto con métricas como la precisión (accuracy) y tasa de error (error_rate), que permitieron un monitoreo constante durante el entrenamiento y facilitaron ajustes necesarios.

La implementación del modelo ResNet-50 se realizó utilizando la biblioteca FastAI, que ofrece herramientas eficientes para la creación y entrenamiento de modelos de aprendizaje profundo. El modelo fue descargado automáticamente desde la biblioteca de PyTorch. En la **Figura 31** se muestra la configuración del modelo y su obtención, tal como aparece en la salida de consola.

```
model_arch = resnet50
learn = vision_learner(dls,
                       model_arch,
                       n_in=1,
                       loss_func=CrossEntropyLossFlat(),
                       metrics=[accuracy, error_rate])

Downloading: "https://download.pytorch.org/models/resnet50-11ad3fa6.pth" to /root/.cache/torch/hub/checkpoin
100%|          | 97.8M/97.8M [00:00<00:00, 172MB/s]
```

Figura 31: Configuración del modelo ResNet-50

En este fragmento de código, se define **model_arch** como la arquitectura de ResNet-50. Luego, se crea el objeto **learn** utilizando la función **vision_learner**, donde se especifican los datos cargados (**dls**), la arquitectura del modelo (**model_arch**), el número de canales de entrada (**n_in=1**), la función de pérdida (**CrossEntropyLossFlat()**) y las métricas a evaluar durante el entrenamiento (**accuracy y error_rate**). Este enfoque permite a FastAI gestionar automáticamente el ciclo de entrenamiento, optimizando el rendimiento en la clasificación de vocalizaciones de aves.

Un paso clave en el ajuste de hiperparámetros fue la búsqueda automática de la tasa de aprendizaje óptima. Una tasa inadecuada podría provocar un entrenamiento lento o impedir la convergencia del modelo. En la **Figura *32*** se presenta el código y la gráfica resultante del método **lr_find()** de FastAI, utilizado para identificar la tasa de aprendizaje ideal.

```
# Encontrar la tasa de aprendizaje ideal
lr = learn.lr_find()[0]/10
print(lr)

6.918309954926372e-05
```

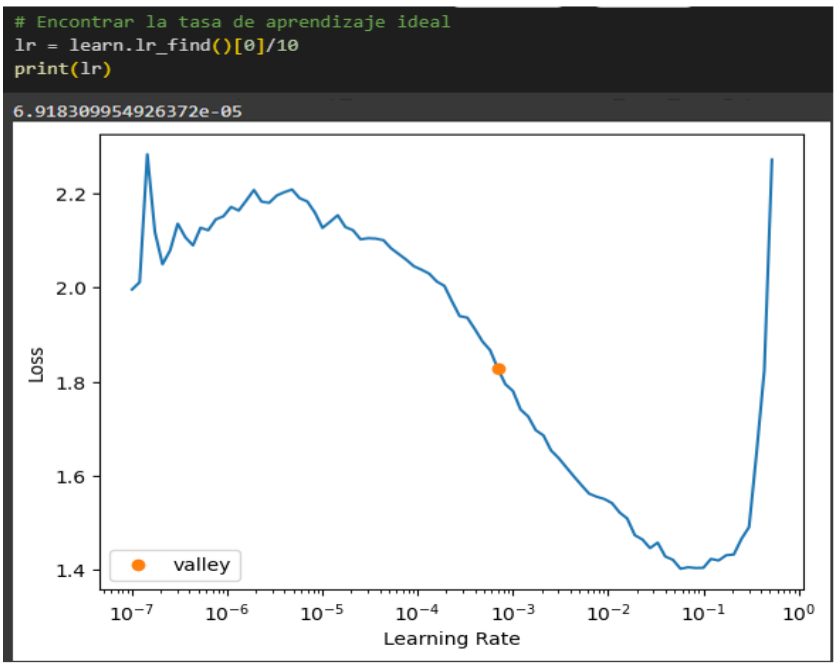


Figura 32: Búsqueda automática de la tasa de aprendizaje

El método lr_find() de Fastai permite explorar distintas tasas de aprendizaje y seleccionar la que minimiza la función de pérdida, identificando un valor óptimo para el entrenamiento. En este caso, se determinó una tasa de aproximadamente 10^{-3}, correspondiente al punto "valley" en la gráfica, donde la pérdida comienza a reducirse significativamente antes de estabilizarse. Este valor equilibra eficiencia y estabilidad, evitando tanto la convergencia lenta con tasas más bajas como inestabilidad con tasas más altas.

Al ajustar estos hiperparámetros, se buscó maximizar la efectividad del modelo, logrando una clasificación más precisa de las vocalizaciones de las especies de aves estudiadas.

6.1.3.2. Tarea 2: Aplicación de Transfer Learning.

El entrenamiento del modelo ResNet-50 mediante transfer learning, incluyó el uso de callbacks para optimizar la tasa de aprendizaje y gestionar aspectos clave, como la detección de sobreajuste y la selección automática del mejor modelo. En la **Figura 33** se muestra el código implementado para este procedimiento.

```
# Callbacks para guardar el mejor modelo y ajuste de la tasa de aprendizaje
callbacks = [
    SaveModelCallback(monitor='accuracy', fname='/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V1'),
    ReduceLROnPlateau(monitor='valid_loss', min_delta=0.1, patience=2),
    EarlyStoppingCallback(monitor='valid_loss', min_delta=0.05, patience=4)
]

# Entrenamiento del modelo
learn.fine_tune(30, wd=0.1, base_lr=lr, cbs=callbacks)
```

Figura 33: Aplicación de callbacks para el entrenamiento

Los callbacks utilizados y sus funciones son los siguientes:

1. **SaveModelCallback**: Este callback se encarga de guardar la versión del modelo con el mejor desempeño en términos de precisión (accuracy) en el conjunto de validación. Se monitorea la métrica accuracy, y cada vez que el modelo alcanza una precisión más alta que la previamente guardada, se reemplaza el modelo almacenado. En este caso, el modelo se guarda en la ruta '/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V1' en el formato .pth, lo que asegura que el mejor modelo esté disponible para su uso o evaluación posterior.
2. **ReduceLROnPlateau**: Este callback ajusta automáticamente la tasa de aprendizaje (learning rate) cuando el desempeño del modelo no mejora. Se monitorea la pérdida en el conjunto de validación (valid_loss), y si no se observa una mejora significativa (definida por min_delta=0.1) durante un número específico de épocas (patience=2), la tasa de aprendizaje se reduce. Esto permite al modelo entrenarse a una tasa más baja cuando se está acercando a una solución óptima, evitando así posibles oscilaciones o inestabilidad en el aprendizaje.

3. **EarlyStoppingCallback**: Este callback detiene el entrenamiento si no se observan mejoras en la pérdida de validación durante un número determinado de épocas consecutivas. Con monitor='valid_loss', min_delta=0.05, y patience=4, se detiene el entrenamiento si, después de cuatro épocas, la pérdida no mejora al menos en 0.05. Esto ayuda a prevenir el sobreentrenamiento del modelo, que podría llevar a un desempeño deficiente en datos no vistos.

El entrenamiento se realizó con el método fine_tune, ajustando el modelo mediante transfer learning durante 30 épocas (**epochs=30**), con una decadencia de peso (**wd=0.1**) y utilizando la tasa de aprendizaje óptima (**base_lr=lr**) previamente identificada. En la **Tabla 7** se detalla el proceso de entrenamiento, iteración por iteración, hasta guardar el mejor modelo.

Tabla 7: Proceso entrenamiento de modelo con tasa de aprendizaje encontrada

epoch	train_loss	valid_loss	acurracy	error_rate	time
0	1.490817	0.944488	0.581481	0.418519	03:14
1	1.435738	0.906674	0.607407	0.392593	03:05
2	1.281671	0.864228	0.622222	0.377778	03:05
3	1.188728	0.725223	0.692593	0.307407	03:24
4	1.084335	0.656335	0.751852	0.248148	03:07
5	0.999800	0.520402	0.807407	0.192593	03:51
6	0.928634	0.515827	0.770370	0.229630	03:08
7	0.865733	0.580878	0.766667	0.233333	03:11
8	0.762196	0.494639	0.822222	0.177778	03:07
9	0.722750	0.448964	0.837037	0.162963	03:13
10	0.660317	0.480913	0.825926	0.174074	03:12
11	0.642926	0.463440	0.818519	0.181481	03:08
12	0.610078	0.388824	0.859259	0.140741	03:27
13	0.565650	0.402907	0.881481	0.118519	03:26
14	0.539397	0.390394	0.840741	0.159259	03:20
15	0.506166	0.339302	0.892593	0.107407	03:23
16	0.477819	0.360540	0.862963	0.137037	03:24

El entrenamiento del modelo finalizó en la época 16, con una duración aproximada de 55 minutos. El proceso se detuvo automáticamente al no observar mejoras en la precisión, evitando el sobreajuste y guardando como mejor modelo el correspondiente a la época 15, que alcanzó una precisión del 89%. Este entrenamiento se realizó en Google Colab, aprovechando el entorno de ejecución gratuito con **T4 GPU** para el desarrollo del proyecto.

En la **Figura 34** se muestra la curva de aprendizaje obtenida durante el entrenamiento.

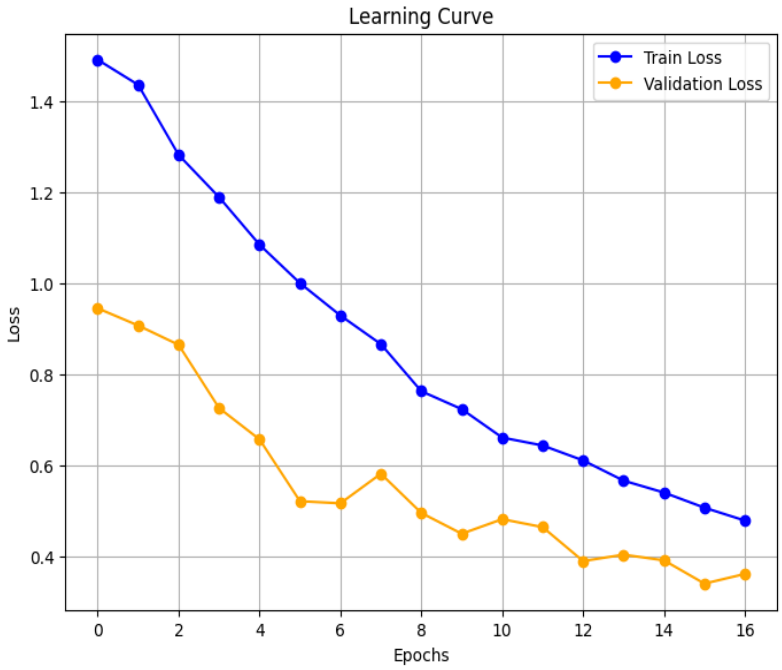


Figura 34: Curva de aprendizaje del entrenamiento con tasa de aprendizaje automática

La curva de aprendizaje muestra que el modelo aprende de manera efectiva en el conjunto de entrenamiento, alcanzando una alta precisión en el conjunto de validación, aunque no perfecta. Esto sugiere que el modelo captura patrones relevantes sin evidencias significativas de sobreajuste. Para mejorar su desempeño y evaluar distintas configuraciones, se desarrollaron varias versiones del modelo utilizando diferentes tasas de aprendizaje, lo que permitió seleccionar la configuración más óptima.

6.1.3.3. Tarea 3: Versionamiento de Modelos.

Además de la versión inicial con una tasa de aprendizaje automática[20] que alcanzó una precisión del 89%, el ingeniero Oscar Cumbicus, experto en machine learning, recomendó comenzar con una tasa de aprendizaje baja. Por ello, se realizaron tres experimentos adicionales con tasas de aprendizaje de 10^{-5} (baja), 10^{-3} (media) y 10^{-1} (alta), lo que permitió comparar los resultados y seleccionar el modelo con mejor desempeño.

Versión 2 con tasa de aprendizaje 10^{-5}

En esta versión[21], se conservaron los mismos callbacks y parámetros que en la configuración inicial, ajustando únicamente el valor de la tasa de aprendizaje en 10^{-5} . En la **Tabla 8** se detalla el proceso de entrenamiento realizado con esta configuración.

[20] Modelo Versión 1
[21] Modelo Versión 2

Tabla 8: Proceso de entrenamiento con tasa de aprendizaje establecida en 10^{-5}

epoch	train_loss	valid_loss	acurracy	error_rate	time
0	1.907428	1.202943	0.455556	0.544444	03:01
1	1.847459	1.227784	0.440741	0.559259	03:00
2	1.767576	1.174323	0.496296	0.503704	03:01
3	1.803918	1.112836	0.466667	0.533333	02:59
4	1.708424	1.155991	0.455556	0.544444	03:00
5	1.620196	1.044523	0.529630	0.470370	03:00
6	1.572418	0.968726	0.566667	0.433333	03:07
7	1.488460	0.840563	0.637037	0.362963	03:03
8	1.461462	0.901862	0.592593	0.407407	03:06
9	1.405166	0.807255	0.670370	0.329630	03:06
10	1.361152	0.865787	0.651852	0.348148	03:02
11	1.343937	0.710334	0.714815	0.285185	03:03
12	1.254428	0.731763	0.692593	0.307407	03:07
13	1.189208	0.778941	0.685185	0.314815	03:02
14	1.198402	0.693480	0.737037	0.262963	03:01
15	1.164303	0.715953	0.718518	0.281481	03:01

Como era de esperarse, el modelo no logró un aprendizaje óptimo con una tasa de aprendizaje baja de 10^{-5}, obteniendo un rendimiento limitado y una precisión final del 73%. Aunque se observa una mejora gradual en la precisión a lo largo de las épocas, el progreso es lento debido a la limitada capacidad de ajuste que proporciona esta tasa. En la **Figura 35,** la curva de aprendizaje refleja este comportamiento, mostrando un aumento progresivo en la precisión, pero sin alcanzar valores elevados. El modelo se estabiliza en una precisión subóptima, lo que indica que el aprendizaje fue insuficiente para capturar completamente los patrones relevantes en los datos.

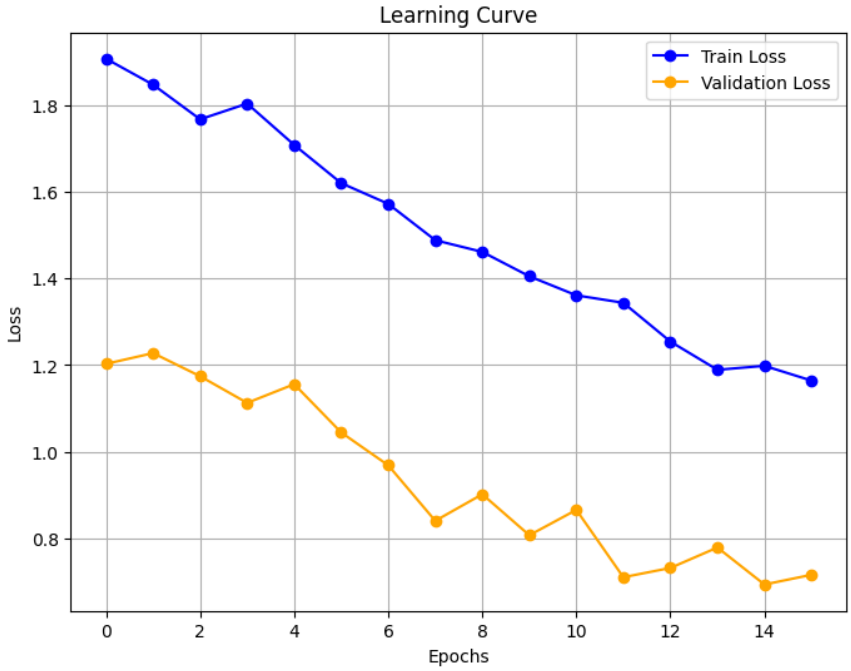


Figura 35: Curva de aprendizaje del entrenamiento con tasa de aprendizaje de 10^{-5}

Versión 3 con tasa de aprendizaje 10^{-3}

Para esta versión[22], se utilizó una tasa de aprendizaje 10^{-3}, manteniendo los callbacks y parámetros definidos en versiones anteriores. En la **Tabla 9** se presentan los resultados del proceso de entrenamiento con esta tasa, destacando su impacto en el desempeño del modelo.

Tabla 9: Proceso de entrenamiento con tasa de aprendizaje establecida en 10^{-3}

epoch	train_loss	valid_loss	acurracy	error_rate	time
0	0.631875	0.452404	0.833333	0.166667	03:00
1	0.539187	0.442922	0.870370	0.129630	02:55
2	0.497626	0.407073	0.855556	0.144444	02:55
3	0.445883	0.400562	0.851852	0.148148	02:54
4	0.420203	0.370734	0.874074	0.125926	02:53
5	0.351904	0.421650	0.877778	0.122222	02:55
6	0.329640	0.412662	0.881481	0.118519	02:54
7	0.265503	0.340784	0.903704	0.096296	02:55
8	0.226349	0.348991	0.888889	0.111111	02:54
9	0.185955	0.325463	0.918519	0.081481	02:57
10	0.194554	0.303739	0.896296	0.103704	02:53
11	0.164132	0.302636	0.900000	0.100000	02:53

[22] Modelo Versión 3

Durante el entrenamiento, el modelo alcanzó una precisión del 91%, demostrando un aprendizaje efectivo de los patrones en los datos. A medida que avanzaron las épocas, se observó una reducción constante en la pérdida y un aumento sostenido en la precisión, evidenciando su capacidad para capturar las características esenciales para la clasificación. En la **Figura 36**, la curva de aprendizaje muestra la pérdida de entrenamiento y validación disminuyendo progresivamente, indicando una mejora consistente sin señales de sobreajuste. La estabilidad alcanzada en las últimas épocas confirma que el modelo logró un buen nivel de generalización, obteniendo una precisión excelente.

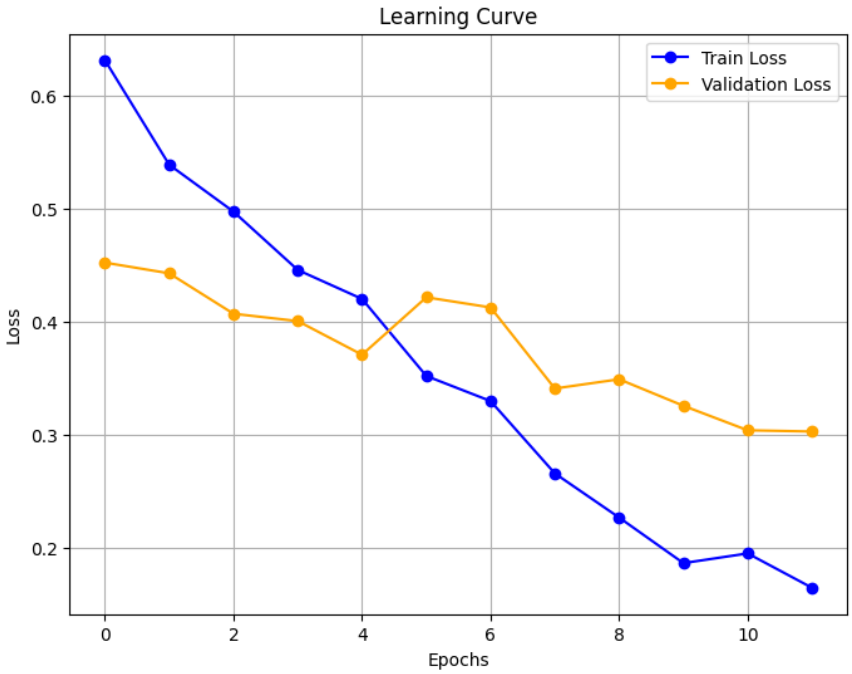


Figura 36: Curva de aprendizaje del entrenamiento con tasa de aprendizaje de 10^{-3}

Versión 4 con tasa de aprendizaje 10^{-1}

En esta versión, se incrementó la tasa de aprendizaje a 10^{-1}, manteniendo la misma configuración de las versiones anteriores. En la **Tabla 10** se muestra el proceso de entrenamiento con esta tasa de aprendizaje elevada.

Tabla 10: Proceso de entrenamiento con tasa de aprendizaje establecida en 10^{-1}

epoch	train_loss	valid_loss	acurracy	error_rate	time
0	1.247278	1.341986	0.811111	0.188889	03:05
1	0.710155	4.216859	0.759259	0.240741	03:02
2	0.593439	0.372917	0.896296	0.103704	03:02
3	0.437750	0.394411	0.925926	0.074074	03:01
4	0.422144	2.571835	0.851852	0.148148	03:04
5	0.785285	7.924121	0.411111	0.588889	02:59
6	1.259747	64.922501	0.662963	0.337037	03:00

El modelo[23] alcanzó una precisión del 92% en la tercera época, superando ligeramente la precisión de la versión anterior (91%). Sin embargo, esta configuración introdujo inestabilidad en el proceso de entrenamiento. A partir de la segunda y cuarta época, la pérdida de validación comenzó a aumentar considerablemente, alcanzando un valor extremo de 64.92 en la sexta época, lo que indica que el modelo empezó a sobreajustarse y perdió capacidad de generalización en los datos de validación. Aunque el modelo mostró un buen rendimiento inicial, la tasa de aprendizaje elevada provocó fluctuaciones en la pérdida de validación, afectando su estabilidad y reduciendo la confiabilidad de la precisión final.

En la **Figura 37** se muestra la curva de aprendizaje con la tasa de aprendizaje 10^{-1}, donde se observa un comportamiento fluctuante entre la pérdida de entrenamiento y la de validación. Mientras que la pérdida de entrenamiento disminuye gradualmente, indicando que el modelo sigue aprendiendo, la pérdida de validación experimenta variaciones abruptas, especialmente, especialmente en ciertas épocas donde se dispara a valores muy altos.

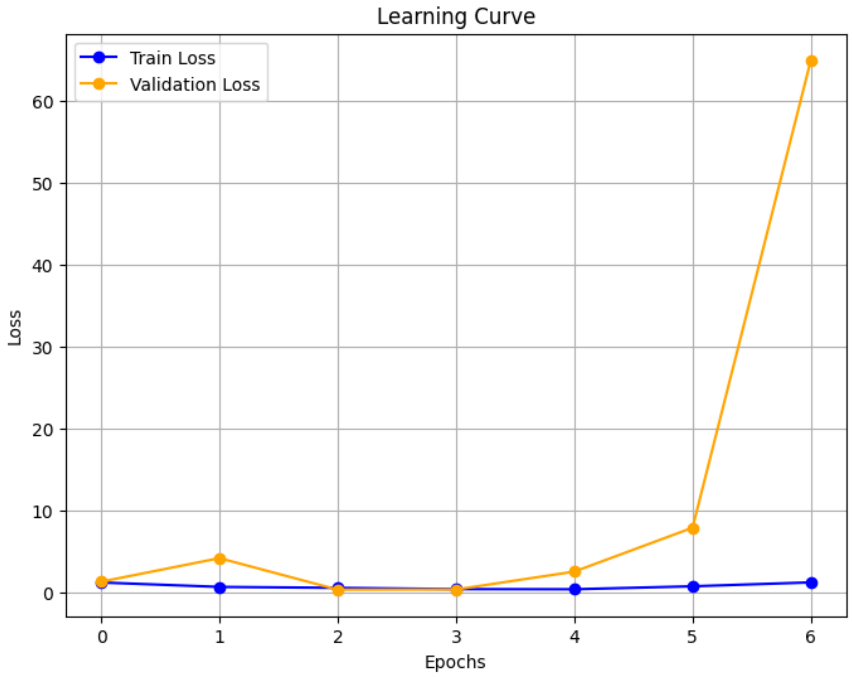


Figura 37: Curva de aprendizaje del entrenamiento con tasa de aprendizaje de 10^{-1}

Conclusión y selección del mejor modelo

En la **Tabla 11** se presenta un resumen de los resultados obtenidos por las diferentes versiones del modelo ResNet-50.

Tabla 11: Resumen de resultados de las versiones del modelo

Versión	Tasa de Aprendizaje	Precisión	Observaciones
V1	Automática	89%	Rendimiento estable, buena generalización, pero no óptimo.

[23] Modelo Versión 4

Versión	Tasa de Aprendizaje	Precisión	Observaciones
V2	10^{-5}	73%	Aprendizaje lento, insuficiente para capturar patrones clave.
V3	10^{-3}	91%	Mejor balance entre precisión, pérdida y estabilidad.
V4	10^{-1}	92%	Alta precisión inicial, pero evidencias de sobreajuste.

De las cuatro versiones entrenadas, la V3 fue seleccionada como el modelo óptimo debido a su equilibrio entre precisión y estabilidad. A pesar de que la versión V4 alcanzó un porcentaje de precisión ligeramente mayor (92%), presentó señales claras de sobreajuste, evidenciadas por un incremento significativo en la pérdida de validación a partir de la cuarta época. Por otro lado, la versión V3 mantuvo una precisión del 91%, con una curva de aprendizaje consistente que muestra una reducción progresiva en la pérdida y una alta capacidad de generalización.

Además, la versión V2, con una tasa de aprendizaje baja (10^{-5}), mostró un rendimiento significativamente inferior (73%), mientras que la versión V1, aunque estable, no alcanzó la precisión lograda por V3. Por estas razones, se determinó que la versión V3 cumple mejor con el primer objetivo de este trabajo, ya que permite ajustar el modelo ResNet-50 de manera efectiva utilizando un conjunto de audios, logrando un rendimiento adecuado sin comprometer la generalización a datos no vistos. En el **Anexo *2***, se presenta el enlace hacia el código de toda la creación del modelo, lo que permite replicar y validar los procesos implementados en este trabajo.

6.2. Objetivo 2: Evaluar el porcentaje global de acierto del modelo entrenado en la clasificación de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

6.2.1. Evaluación de modelos de aprendizaje automático

6.2.1.1. Tarea 1: Validación del rendimiento del modelo.

En esta tarea, se analiza la matriz de confusión del mejor modelo seleccionado, la versión V3. Dicha matriz refleja el rendimiento del modelo en la clasificación de audios de las tres especies de aves estudiadas, destacando tanto las predicciones correctas como los errores de clasificación entre las clases. Este análisis es esencial para evaluar la capacidad del modelo en términos de generalización y precisión. En la **Figura 38** se muestra la matriz de confusión generada para la versión V3.

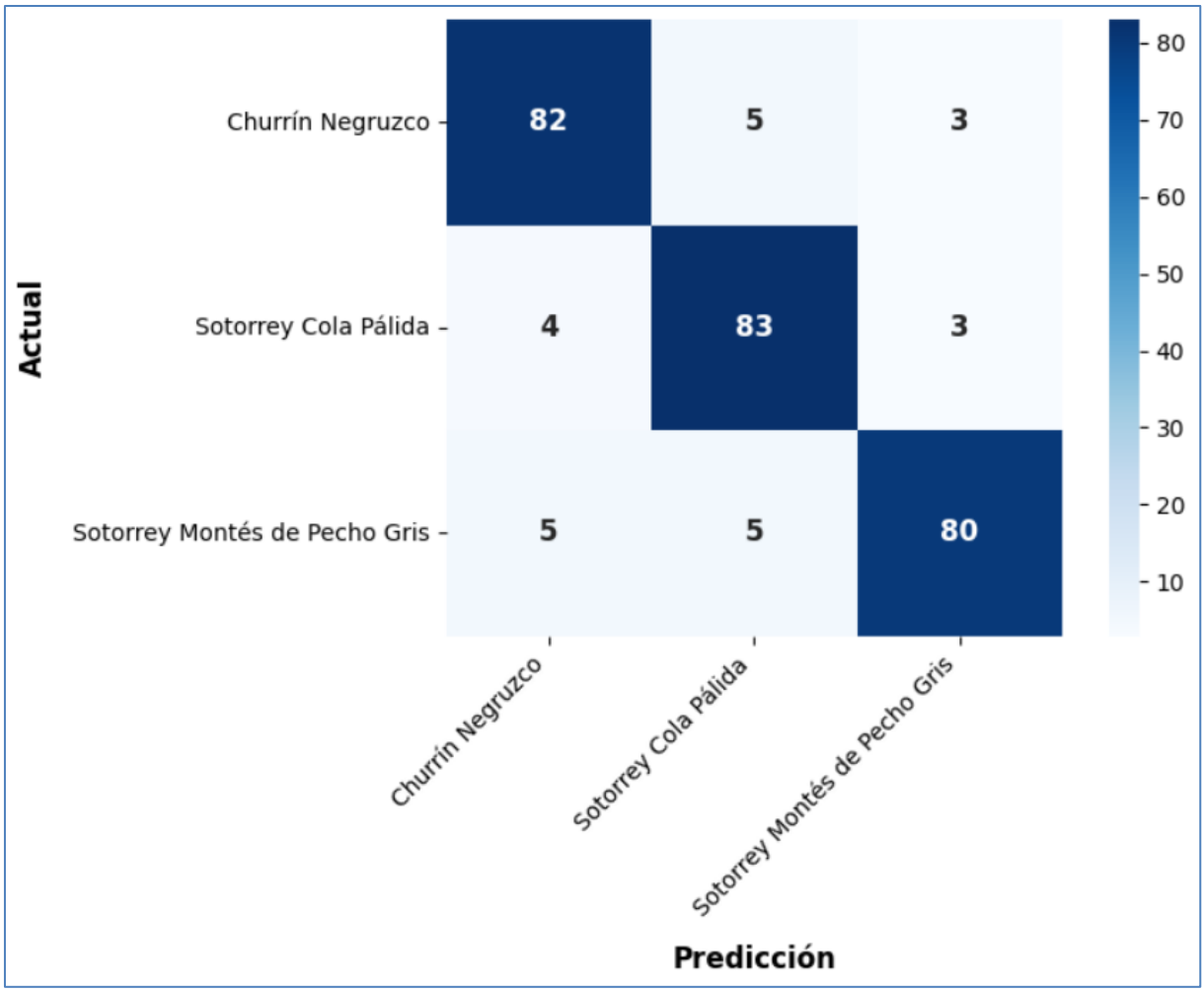


Figura 38: Matriz de confusión del modelo versión V3

La matriz de confusión destaca que la mayoría de las predicciones son correctas, con 82 clasificaciones correctas para “Churrín Negruzco”, 83 para “Soterrey Cola Pálida” y 80 para “Soterrey Montés de Pecho Gris”. Estos resultados evidencian la capacidad del modelo para identificar con precisión las especies estudiadas. No obstante, se identificaron algunos errores de clasificación: 5 confusiones entre “Soterrey Montés de Pecho Gris” y “Soterrey Cola Pálida” y 3 confusiones entre “Churrín Negruzco” y “Soterrrey Montés de Pecho Gris”. A pesar de estas limitaciones, el modelo muestra un rendimiento destacable para distinguir entre las clases.

En la **Tabla 12** se detallan los valores de los componentes de la matriz de confusión para cada especie de ave: TP (Verdaderos Positivos), FP (Falsos Positivos) y FN (Falsos Negativos). Estos valores proporcionan información determinante para evaluar el desempeño del modelo en la identificación de las vocalizaciones de las especies estudiadas, destacando tanto los aciertos como los errores de clasificación.

Tabla 12: Valores de los componentes de la matriz de confusión por clase

Clase	TP	FP	FN
Churrín Negruzco	82	5	3
Soterrey Cola Pálida	83	4	3
Soterrey Montés de Pecho Gris	80	5	5

Cálculo de las métricas

Dado que la métrica principal de este proyecto es la precisión, se ha priorizado su cálculo y análisis. Además, se presenta el porcentaje global de acierto (accuracy) como métrica complementaria para ofrecer una visión general del rendimiento del modelo.

A continuación, se presentan los cálculos manuales de las métricas más relevantes para evaluar el rendimiento del modelo:

Precisión:

$$\boldsymbol{Churrín\ Negruzco} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} = \frac{\boldsymbol{82}}{\boldsymbol{82 + 5}} = \boldsymbol{0.94}\ (\boldsymbol{94\%})$$

$$\boldsymbol{Soterrey\ Cola\ Pálida} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} = \frac{\boldsymbol{83}}{\boldsymbol{83 + 4}} = \boldsymbol{0.95}\ (\boldsymbol{95\%})$$

$$\boldsymbol{Soterrey\ Montés\ de\ Pecho\ Gris} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} = \frac{\boldsymbol{80}}{\boldsymbol{80 + 5}} = \boldsymbol{0.94}\ (\boldsymbol{94\%})$$

Porcentaje global de acierto:

$$\boldsymbol{Acierto} = \frac{\boldsymbol{82 + 83 + 80}}{\boldsymbol{270}} = \frac{\boldsymbol{245}}{\boldsymbol{270}} = \boldsymbol{0.9074}\ (\boldsymbol{90.74\%})$$

Tasa de error:

$$\boldsymbol{Tasa\ de\ error} = \frac{\boldsymbol{(5 + 3 + 4 + 3 + 5 + 5)}}{\boldsymbol{270}} = \frac{\boldsymbol{25}}{\boldsymbol{270}} = \boldsymbol{0.0926}\ (\boldsymbol{9.26\%})$$

Sensibilidad:

$$\boldsymbol{Churrín\ Negruzco} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} = \frac{\boldsymbol{82}}{\boldsymbol{82 + 3}} = \boldsymbol{0.96}\ (\boldsymbol{96\%})$$

$$\boldsymbol{Soterrey\ Cola\ Pálida} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} = \frac{\boldsymbol{83}}{\boldsymbol{83 + 3}} = \boldsymbol{0.97}\ (\boldsymbol{97\%})$$

$$\boldsymbol{Soterrey\ Montés\ de\ Pecho\ Gris} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} = \frac{\boldsymbol{80}}{\boldsymbol{80 + 5}} = \boldsymbol{0.94}\ (\boldsymbol{94\%})$$

F1-Score:

$$\boldsymbol{Churrín\ Negruzco} = \boldsymbol{2 * \frac{P * S}{P + S} = 2 * \frac{0.94 * 0.96}{0.94 + 0.96} = 0.95}\ (\boldsymbol{95\%})$$

$$\boldsymbol{Soterrey\ Cola\ Pálida} = \boldsymbol{2 * \frac{P * S}{P + S} = 2 * \frac{0.95 * 0.97}{0.95 + 0.97} = 0.96}\ (\boldsymbol{96\%})$$

$$\boldsymbol{Soterrey\ Montés\ de\ Pecho\ Gris} = \boldsymbol{2 * \frac{P * S}{P + S} = 2 * \frac{0.94 * 0.94}{0.94 + 0.94} = 0.94}\ (\boldsymbol{94\%})$$

Los cálculos manuales realizados confirman los valores reportados previamente, validando el buen rendimiento del modelo en esta tarea de clasificación.

Evaluación del modelo con datos de prueba

El desempeño del modelo entrenado se evaluó utilizando el conjunto de datos de prueba. Se configuraron un Datablock y un DataLoader específicos para estas muestras, aplicando las transformaciones necesarias y empleando la arquitectura ResNet-50. Tras cargar el modelo preentrenado almacenado, se obtuvieron las predicciones del conjunto de prueba en dos partes y se calculó una precisión combinada de 89.33% (**TensorBase(0.8933)**), este valor refleja la capacidad del modelo para identificar correctamente las especies de aves en el conjunto de datos no visto durante el entrenamiento.

En la **Figura 39** se presenta el fragmento de código utilizado para implementar el proceso de evaluación y obtener el porcentaje de precisión del modelo.

```
# Directorio de prueba
test_dir = path / 'complete_dataset/test'

# Crear DataBlock para los datos de prueba
auds_test = DataBlock(blocks=(AudioBlock(crop_signal_to=seconds*1000), CategoryBlock),
                      get_x=ColReader("filename", pref=test_dir),
                      item_tfms=item_transforms,
                      get_y=ColReader("category"))

# Crear DataLoader para los datos de prueba
dls_test = auds_test.dataloaders(df_test, bs=32, shuffle=False, seed=42)

model_arch = resnet50
learn3 = vision_learner(dls_test,
            model_arch,
            n_in=1,
            loss_func=CrossEntropyLossFlat(),
            metrics=[accuracy, error_rate])
learn3.load('/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V3')
dls_test.rng.seed(42)
set_seed(42,True)

preds1, y = learn3.get_preds(dl=dls_test[0])
acc1 = accuracy(preds1, y)

preds2, y = learn3.get_preds(dl=dls_test[1])
acc2 = accuracy(preds2, y)

print((acc1*preds1.shape[0] + acc2*preds2.shape[0])/(preds1.shape[0] + preds2.shape[0]))
```

```
/usr/local/lib/python3.10/dist-packages/fastai/learner.py:53: FutureWarning: You are using `tor
  state = torch.load(file, map_location=device, **torch_load_kwargs)
/usr/local/lib/python3.10/dist-packages/fastai/learner.py:61: UserWarning: Saved file doesn't c
  elif with_opt: warn("Saved file doesn't contain an optimizer state.")
TensorBase(0.8933)
```

Figura 39: Código de evaluación del modelo con datos de prueba

La precisión de 89.33% refleja un desempeño sobresaliente del modelo, logrando clasificar correctamente la mayoría de las grabaciones del conjunto de prueba. Este resultado sugiere que el modelo generaliza adecuadamente y mantiene un equilibrio entre los aciertos y errores, gracias a una configuración adecuada de parámetros y técnicas de preprocesamiento empleadas durante el entrenamiento.

6.2.1.2. Tarea 2: Empaquetado del modelo.

En esta tarea, se exporta el mejor modelo seleccionado (versión V3) desde el formato **.pth** a **.pkl.** La elección del formato **.pkl** se debe a que permite empaquetar tanto el modelo como sus configuraciones, facilitando su posterior uso en aplicaciones externas y asegurando su compatibilidad con diferentes entornos de despliegue. El formato **.pth** solo almacena los pesos del modelo. En la **Figura 40**, se presenta el fragmento de código que realiza la operación de exportación del modelo, esto se implementó utilizando la herramienta FastAI, que ofrece métodos como **learn.load** y **learn.export.**

```
learn.load('/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V3')
learn.export('/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V3.pkl')
```

Figura 40: Código para la exportación del modelo al formato .pkl

Uso del modelo empaquetado para la clasificación de audios externos

El modelo empaquetado en formato **.pkl**[24] se implementa en un pipeline diseñado para la predicción de audios externos[25]. Estos audios fueron recopilados en bibliotecas especializadas en el estudio de aves, diferentes a Xeno-canto, con el propósito de evaluar la capacidad del modelo para clasificar correctamente grabaciones no incluidas en los conjuntos de datos originales del proyecto. Este pipeline abarca todas las etapas del procesamiento, desde la carga del archivo de audio hasta la generación del espectrograma y la predicción final de la especie de ave correspondiente.

En la **Figura 41**, se muestra el código empleado para realizar la inferencia. Este pipeline es flexible, ya que permite modificar únicamente la ruta del audio para clasificar diferentes grabaciones, manteniendo el resto del flujo de procesamiento sin cambios.

[24] Modelo empaquetado V3
[25] Audios externos

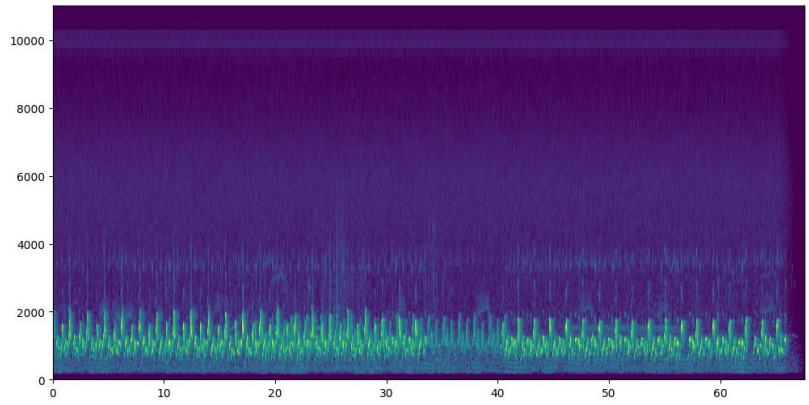
```python
import librosa
import pandas as pd
import fastaudio.core.signal as fas
from fastai.vision.all import load_learner

def clasificar_audio(ruta_audio):
    # Cargar el modelo
    modelo = load_learner('/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V3.pkl', cpu=True)

    # Cargar y preprocesar el audio
    y, sr = librosa.load(ruta_audio, sr=None)  # Especificar sr=None para mantener la tasa de muestreo original
    espectrograma = librosa.feature.melspectrogram(y=y, sr=sr)

    # Convertir el espectrograma a un DataFrame de pandas
    df_espectrograma = pd.DataFrame(data=espectrograma)

    # Convertir el DataFrame a un AudioTensor de fastaudio
    tensor_espectrograma = fas.AudioTensor(df_espectrograma.values, sr=sr)  # Especificar la tasa de muestreo

    # Inferencia
    prediccion, _, _ = modelo.predict(tensor_espectrograma)

    # Obtener la especie predicha
    especie_predicha = prediccion

    return especie_predicha

# Obtener la ruta del audio
ruta_audio = '/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Churrín Negruzco.mp3'


# Realizar la clasificación
especie_predicha = clasificar_audio(ruta_audio)

print(f"La especie de ave predicha es: {especie_predicha}")
```

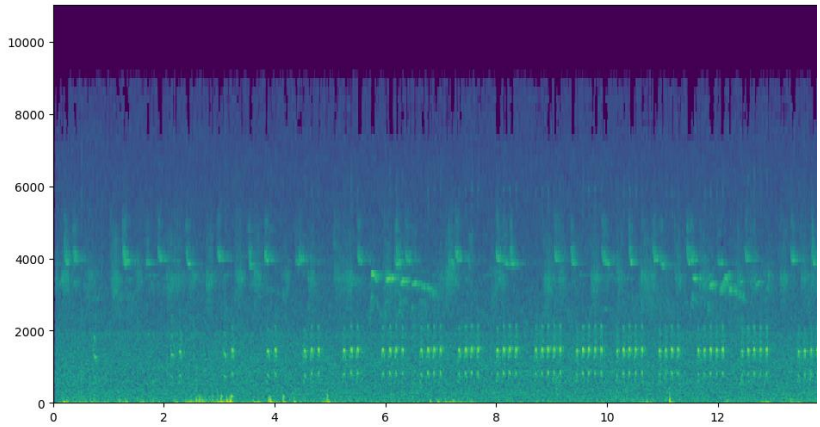
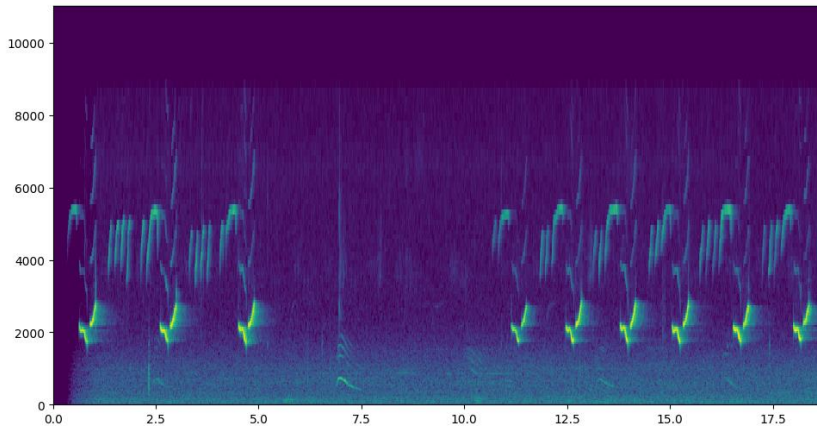
```
La especie de ave predicha es: Churrín Negruzco
```

Figura 41: Pipeline para la clasificación de audios externos

En esta inferencia se utilizó el archivo “Audio Churrín Negruzco”, por lo que el modelo predijo correctamente esta misma especie de ave. En la **Tabla 13**, se presenta un resumen para los siguientes audios procesados, el cual contiene la ruta, los espectrogramas generados por cada archivo y la especie identificada por el modelo. Este enfoque permite analizar visualmente las características del audio que contribuyen a la predicción.

Tabla 13: Resumen de inferencias y espectrogramas generados

Ruta del audio	Espectrograma	Predicción
'/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Churrín Negruzco.mp3'		Churrín Negruzco

Ruta del audio	Espectrograma	Predicción
'/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Soterrey Cola Pálida.mp3'		Soterrey Cola Pálida
'/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Soterrey Montés de Pecho Gris.mp3'		Soterrey Montés de Pecho Gris

La implementación de este pipeline destaca la versatilidad del modelo empaquetado, demostrando su capacidad para procesar diferentes audios y generalizar correctamente a nuevas grabaciones externas.

Uso del modelo empaquetado para la clasificación simultánea de múltiples audios externos

En esta sección, se implementa un pipeline para clasificar múltiples audios externos de forma simultánea, utilizando el modelo empaquetado en formato **.pkl.** Este proceso no solo identifica la especie predicha para cada audio, sino que también proporciona las probabilidades de pertenencia a cada clase, permitiendo un análisis más detallado del rendimiento del modelo.

En la **Figura 42**, se presenta el código que carga el modelo empaquetado con load_learner, procesa los audios con Librosa para generar espectrogramas, clasifica cada

archivo obteniendo las probabilidades de cada clase y guarda los resultados en un archivo CSV denominado **resultados_predicciones.csv**[26].

```python
def clasificar_audio(ruta_audio, modelo):
    y, sr = librosa.load(ruta_audio, sr=None)
    espectrograma = librosa.feature.melspectrogram(y=y, sr=sr)
    df_espectrograma = pd.DataFrame(data=espectrograma)
    tensor_espectrograma = fas.AudioTensor(df_espectrograma.values, sr=sr)
    prediccion, _, probs = modelo.predict(tensor_espectrograma)
    return prediccion, probs.numpy()  # Devuelve todas las probabilidades

# Cargar el modelo una sola vez
modelo = load_learner('/content/drive/MyDrive/9no Ciclo/Detección de aves/Models/new_model_resnet50V3.pkl', cpu=True)
import os
# Lista de rutas de audio para clasificar
rutas_audio = [
    '/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Churrín Negruzco.mp3',
    '/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Soterrey Cola Pálida.mp3',
    '/content/drive/MyDrive/9no Ciclo/Detección de aves/Audios externos/Audio Soterrey Montés de Pecho Gris.mp3'
]

# Realizar clasificaciones y guardar resultados
resultados = []
for ruta in rutas_audio:
    nombre_archivo = os.path.basename(ruta)
    especie_predicha, probs = clasificar_audio(ruta, modelo)
    resultados.append([nombre_archivo, str(especie_predicha)] + probs.tolist())

# Actualiza las cabeceras del CSV
cabeceras = ["Archivo", "Especie Predicha"] + [f"Prob_{clase}" for clase in modelo.dls.vocab]

# Ruta donde se guardará el archivo CSV
ruta_csv = '/content/drive/MyDrive/9no Ciclo/Detección de aves/Evaluation/resultados_predicciones.csv'

# Guardar resultados en el archivo CSV
with open(ruta_csv, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(cabeceras)
    writer.writerows(resultados)

print(f"Resultados guardados en '{ruta_csv}'")
```

```
Resultados guardados en '/content/drive/MyDrive/9no Ciclo/Detección de aves/Evaluation/resultados_predicciones.csv'
```

Figura 42: Pipeline para la clasificación simultánea de múltiples audios externos

En la **Tabla 14**, se presentan los resultados generados por el pipeline. La tabla abarca el nombre del archivo de audio, la especie identificada por el modelo y las probabilidades asociadas a cada clase.

Tabla 14: Resultados de clasificación de múltiples audios externos

Archivo	Especie Predicha	Prob_Churrín Negruzco	Prob_Soterrey Cola Pálida	Prob_Soterrey Montés de Pecho Gris
Audio Churrín Negruzco.mp3	Churrín Negruzco	91.902775	0.036393	0.368469
Audio Soterrey Cola Pálida.mp3	Soterrey Cola Pálida	0.079246	91.599028	0.340802
Audio Soterrey Montés de Pecho Gris.mp3	Soterrey Montés de Pecho Gris	0.486207	0.009129	90.093274

[26] CSV resultados_predicciones

La probabilidad alta asociada a la especie identificada en cada caso refleja la confianza del modelo en sus resultados, lo que valida su capacidad para generalizar a grabaciones no incluidas en los conjuntos de datos originales. Este enfoque también proporciona información detallada sobre las probabilidades de cada clase, permitiendo un análisis más profundo del rendimiento del modelo y evidenciando su utilidad en aplicaciones prácticas.

Uso del modelo empaquetado dentro de una aplicación web

En este apartado se describe la implementación del modelo empaquetado en un sitio web desarrollado con Flask, utilizando una estructura organizada que permite gestionar los recursos necesarios para la inferencia y visualización. En la **Figura 43**, se presenta la estructura del proyecto, que incluye las siguientes carpetas y archivos principales. Además, en el **Anexo *3***, se especifican los detalles técnicos del modelo empaquetado, destacando las configuraciones, requerimientos y funcionalidades que permiten su integración y uso eficiente en aplicaciones.

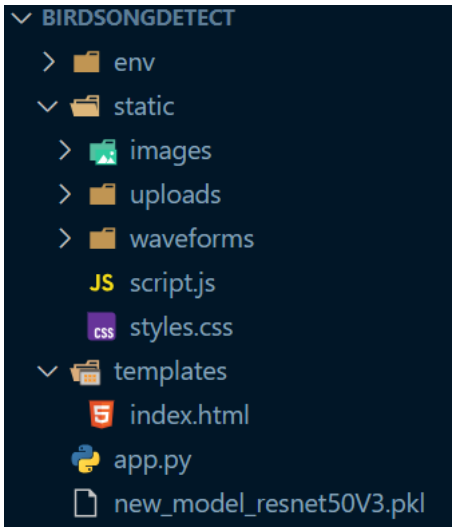


Figura 43: Estructura del proyecto de la aplicación web

- **env:** Entorno virtual que contiene todas las dependencias necesarias para ejecutar el proyecto.
- **static:** Carpeta que almacena recursos estáticos utilizados en la aplicación web, con las siguientes subcarpetas:
    - **images:** Contiene una imagen representativa de cada especie de ave para mostrar durante la ejecución.
    - **uploads:** Almacena todos los archivos de audio que han sido subidos por los usuarios.
    - **waveforms:** Guarda las formas de onda y espectrogramas generados a partir de los audios cargados.

    - **script.js**: Archivo JavaScript que implementa las funciones y acciones asociadas a los botones, barra de progreso y otros elementos interactivos de la interfaz.
    - **styles.css**: Archivo CSS que define el diseño y los estilos de cada elemento de la interfaz web.
- **templates:** Carpeta que contiene el archivo HTML:
    - **index.html:** Plantilla principal de la interfaz web.
- **app.py:** Archivo principal que ejecuta la apliación web y cordina la lógica entre la interfaz y el modelo empaquetado.
- **new_model_resnet50V3.pkl:** Archivo que contiene el modelo empaquetado para realizar las predicciones.

Una vez ejecutado el archivo app.py, se inicia la aplicación dentro de un servidor interno generado por Flask. En la **Figura 44**, se muestra la interfaz principal diseñada para la interacción con el modelo de predicción. Esta interfaz incluye los siguientes elementos destacados:

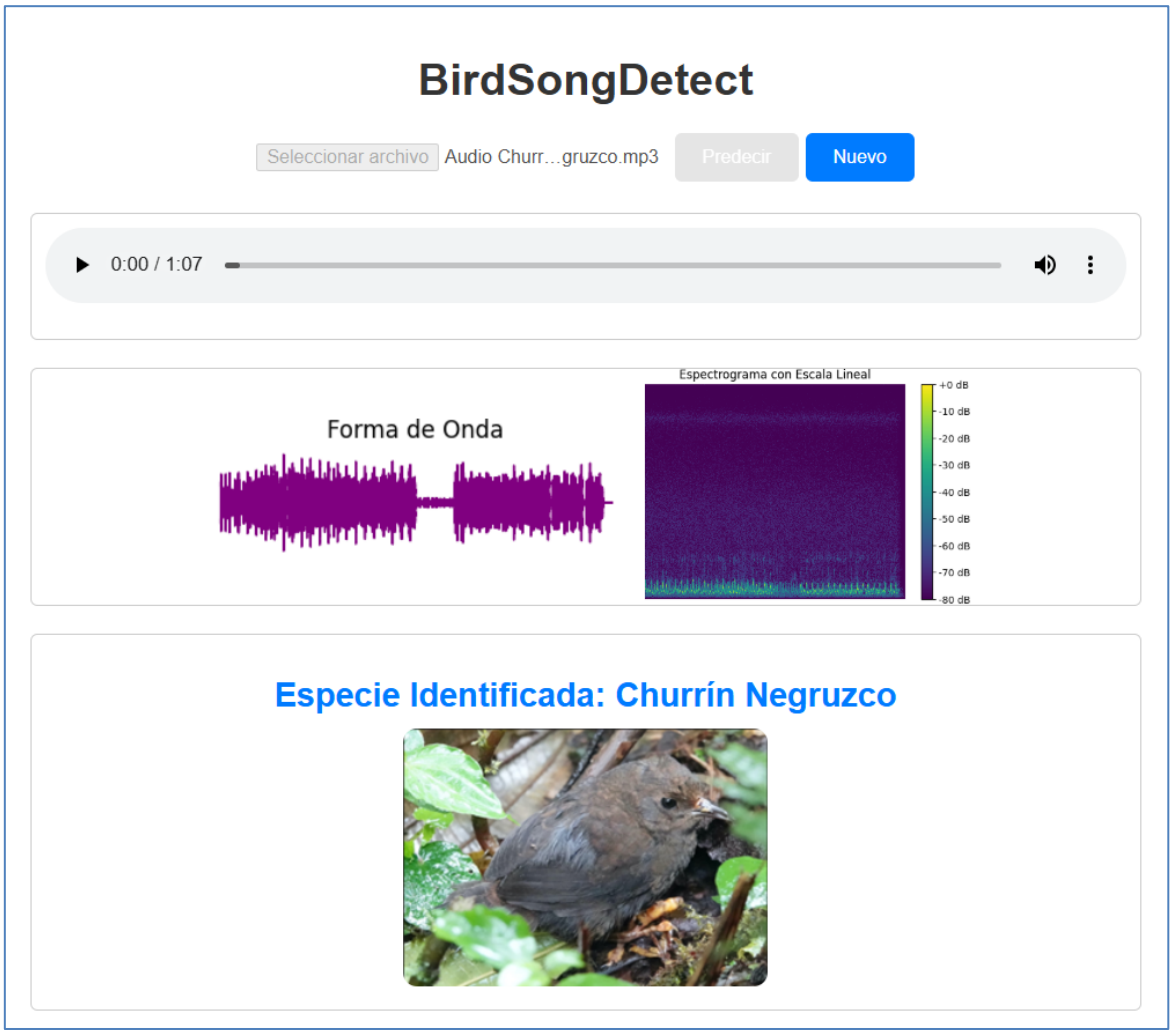


Figura 44: Interfaz web con un ejemplo de predicción exitosa para la especie Churrín Negruzco

- **Título:** El encabezado **BirdSongDetect** representa la marca conceptual para este proyecto.
- **Botón de selección de archivo:** Permite al usuario cargar un archivo de audio desde su administrador de archivos. En este ejemplo se seleccionó el archivo (**Audio Churrín Negruzco.mp3**).
- **Botón "Predecir":** Este botón activa el modelo empaquetado para realizar la inferencia y predecir la especie correspondiente al audio cargado.
- **Botón "Nuevo":** Permite reiniciar el proceso, habilitando nuevamente la selección de un nuevo archivo de audio y borrando los resultados previos,

Cuando se hace click en el botón **Predecir,** se muestra una barra de progreso con la etiqueta "Cargado…" mientras el modelo realiza la predicción. Debajo de esta, se encuentra un reproductor de audio que se habilita automáticamente tras cargar un archivo válido. Con esta información adicional, la interfaz presenta en un recuadro dos gráficos tanto la forma de onda y el espectrograma. Finalmente, en el recuadro inferior, se despliega el nombre de la especie identificada, acompañado de una imagen representativa. En este ejemplo, el modelo predice la especie **Churrín Negruzco,** mostrando su nombre y una fotografía relacionada como resultado final.

Con la predicción exitosa de la especie **Churrín Negruzco**, el modelo demuestra su capacidad para clasificar audios correctamente. De manera similar, se realizaron predicciones exitosas para las demás especies. En la **Figura 45**, se presentan los resultados de la predicción exitosa para la especie **Soterrey Cola Pálida**, mostrando su nombre y su imagen representativa.

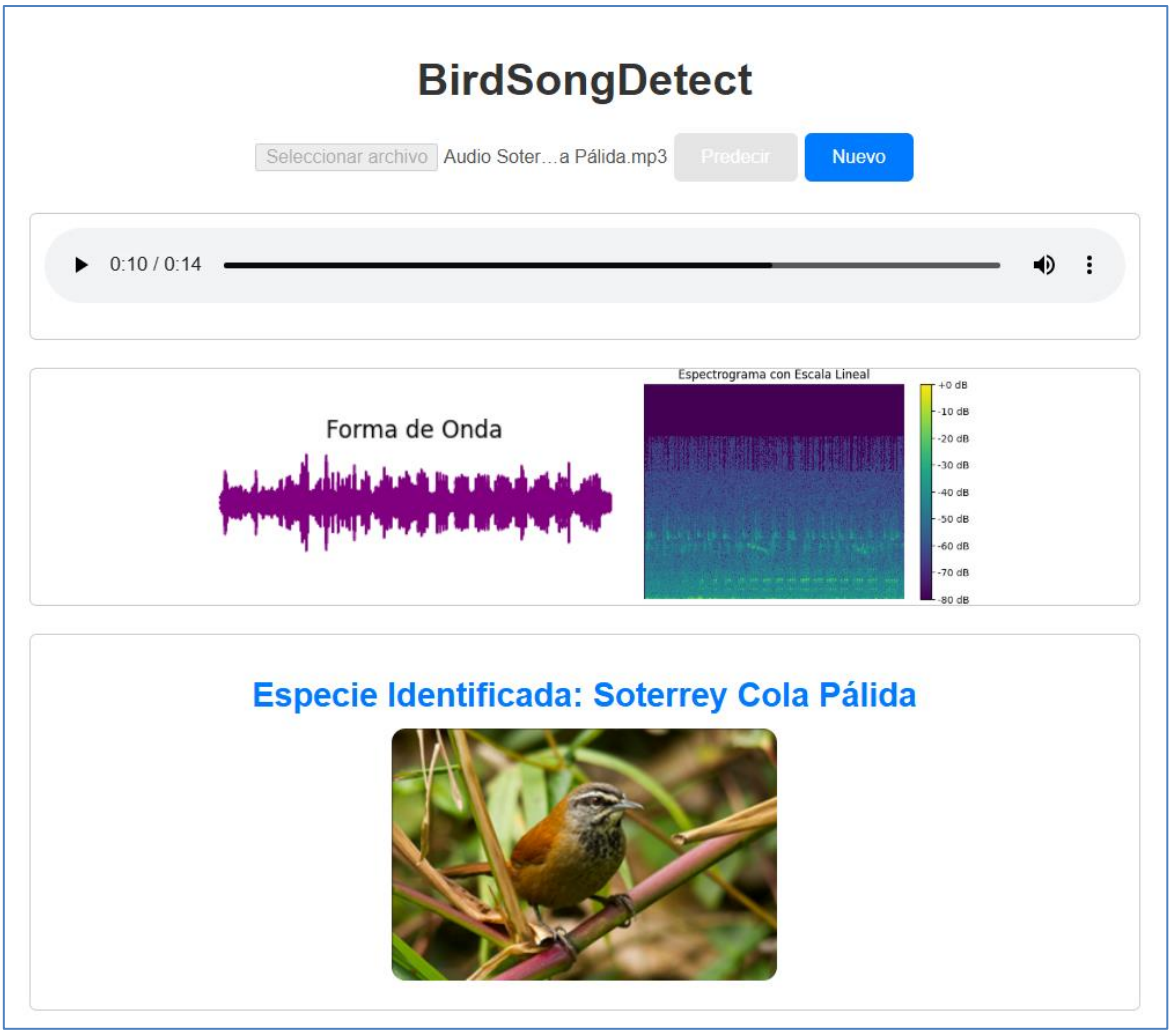


Figura 45: Interfaz web con un ejemplo de predicción exitosa para la especie Soterrey Cola Pálida

En la **Figura 46**, se muestran los resultados correspondientes a la especie Soterrey Montés de Pecho Gris, destacando nuevamente la precisión del modelo.

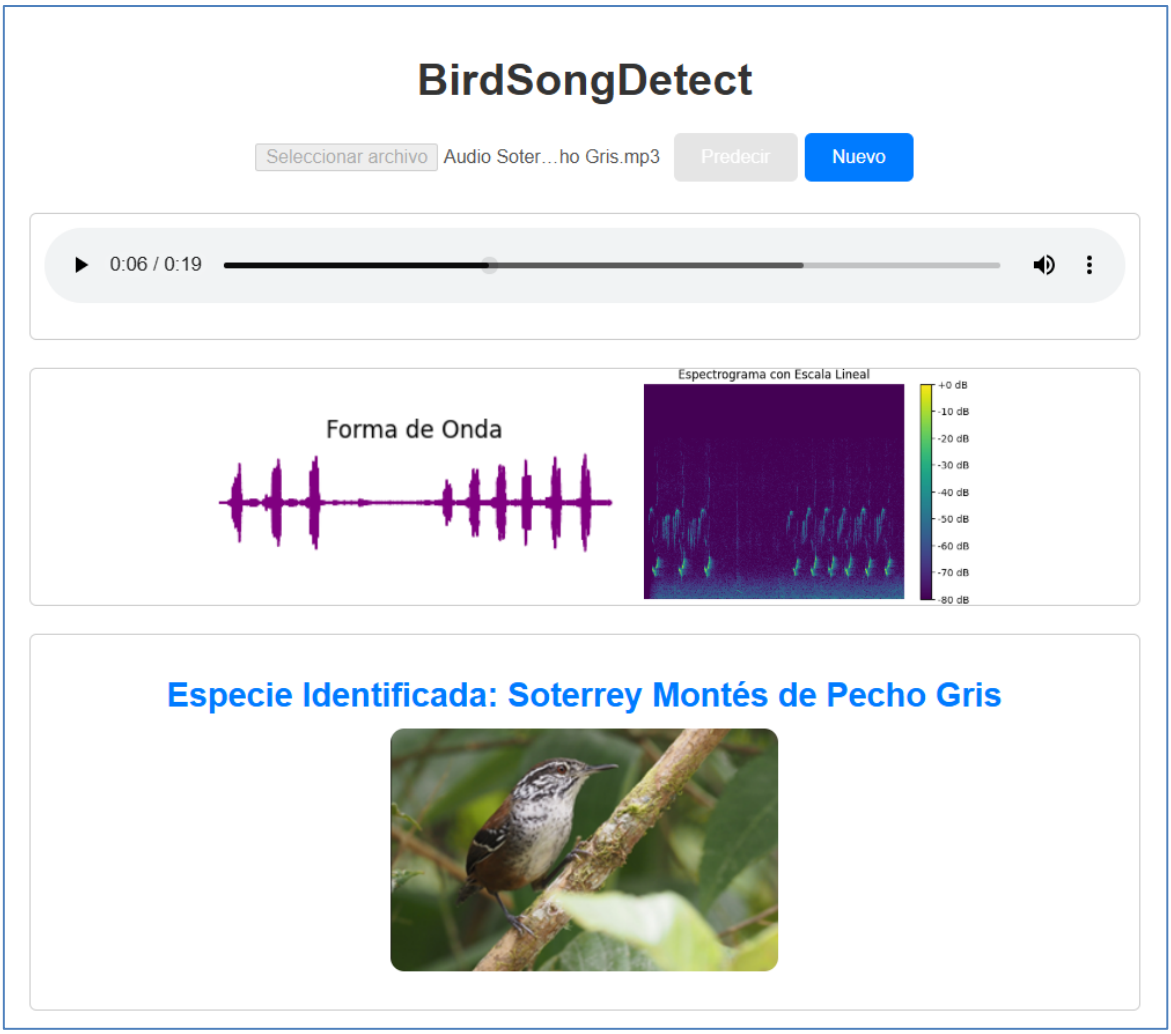


Figura 46: Interfaz web con un ejemplo de predicción exitosa para la especie Soterrey Montés de Pecho Gris

La implementación de la interfaz web demuestra la efectividad y usabilidad del modelo empaquetado, permitiendo clasificar de manera precisa las vocalizaciones de las especies estudiadas. La validación del modelo con audios externos mostró un porcentaje de acierto del 91.90% para Churrín Negruzco, 91.59% para Soterrey Cola Pálida y 90.09% para Soterrey Montés de Pecho Gris. En estas pruebas, el sistema identificó correctamente la especie con un error marginal asociado a la variabilidad de calidad a las grabaciones y la posible superposición de vocalizaciones en algunos audios. Como resultado, el porcentaje global de acierto final del modelo fue del 91.19%, calculado como la media de las precisiones por clase en la validación externa.

Para reforzar su aplicabilidad en contextos reales, la interfaz web incluyó elementos visuales como la forma de onda, el espectrograma y la imagen representativa de la especie identificada, lo que facilitó la interpretación de los resultados y mejoró la experiencia del usuario (ver **Anexo *4***). Este anexo presenta una secuencia de imágenes que ilustran este proceso en detalle, mostrando paso a paso desde la carga del audio hasta la visualización de los resultados, permitiendo una mejor comprensión de su funcionamiento.

7. Discusión

En esta sección se analizan los resultados obtenidos durante el desarrollo del TIC, considerando los dos objetivos específicos planteados. Estos objetivos estuvieron alineados con el objetivo general, abordando tanto el ajuste del modelo como la evaluación de su desempeño en un entorno práctico. A continuación, se detalla la discusión en torno a cada objetivo.

7.1 Primer objetivo: Ajustar el modelo ResNet-50 con un conjunto de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

La metodología CRISP-ML(Q) se destacó como un marco de trabajo flexible permitiendo realizar ajustes iterativos en cada fase del desarrollo del modelo. Este enfoque se diferencia de trabajos como TR01 y TR04, donde se emplearon metodologías más estructuradas pero menos adaptativas, lo que limitó su capacidad para integrar mejoras continuas.

El uso de espectrogramas como representación visual de los audios fue determinante para aprovechar las capacidades del modelo preentrenado ResNet-50, alineándose con estudios como TR02 y TR05 que también destacaron la efectividad de esta técnica en la clasificación de datos bioacústicos. Sin embargo, este proyecto fue más allá al optimizar los parámetros de generación de espectrogramas, obteniendo patrones más consistentes y adecuados para el aprendizaje del modelo.

En comparación con TR03 y TR06, que enfrentaron problemas de desbalance de clases y calidad inconsistente en los datasets, este proyecto empleó técnicas avanzadas, como la unificación de la tasa de muestreo a 44,100 Hz, la normalización de amplitud y el aumento de datos. Estas estrategias permitieron generar un conjunto de datos balanceado y representativo, abordando los desafíos de la variabilidad en la calidad de los audios. Adicionalmente, las estrategias de preprocesamiento, como la conversión de formato y la selección cuidadosa de grabaciones, resultaron esenciales para minimizar errores durante el entrenamiento del modelo, superando limitaciones observadas en TR08 y TR09.

La implementación de la transferencia de aprendizaje fue un elemento diferenciador que permitió superar la limitación de datos, optimizando recursos computacionales y mejorando el rendimiento del modelo. Esto contrasta con estudios como TR07, donde la falta de técnicas avanzadas resultó en una menor precisión del modelo. En este caso, el uso de ResNet-50 y el ajuste de hiperparámetros, como la tasa de aprendizaje y el tamaño de lote, junto con el uso de callbacks, contribuyó a evitar problemas de sobreajuste y a maximizar el rendimiento del modelo, alcanzando una precisión del 91% en el conjunto de entrenamiento y del 89.33% en el conjunto de prueba.

7.2 Segundo objetivo: Evaluar el porcentaje global de acierto del modelo entrenado en la clasificación de audios de las aves Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.

La evaluación del modelo se realizó integrándolo en una interfaz web funcional, permitiendo predicciones sobre audios externos no incluidos en el conjunto de datos original. Este enfoque representa una mejora significativa respecto a trabajos como TR02 y TR05, que se limitaron a evaluaciones en entornos controlados. La capacidad del modelo para generalizar con datos externos, incluyendo grabaciones de diferentes fuentes, validó su aplicabilidad en contextos reales.

El empaquetado del modelo en formato .pkl facilitó su implementación y redujo las barreras técnicas para su utilización, destacándose frente a estudios como TR09, donde no se exploró la integración en aplicaciones prácticas. Además, la interfaz web incluyó elementos visuales como la forma de onda, el espectrograma y la imagen de la especie identificada, mejorando la interpretación de los resultados y aumentando la confianza de los usuarios en el sistema. Estas mejoras posicionan este trabajo por encima de TR06 y TR10, que carecieron de herramientas visuales integradas.

El modelo alcanzó un porcentaje global de acierto del 91.19% cumpliendo con las expectativas planteadas al inicio del proyecto y superando estudios previos como TR05 (88.3%) y TR08 (90.5%). Estos resultados también reflejan la robustez del modelo al manejar datos variados y condiciones desafiantes, como las diferencias en la calidad de audios. Sin embargo, se identificaron limitaciones relacionadas con la dependencia de un conjunto de datos limitado en tamaño, lo que podría mejorarse en investigaciones futuras mediante la inclusión de grabaciones adicionales y la aplicación de técnicas de aprendizaje no supervisado. También se podría considerar la eliminación de ruido y la separación de fuentes en audios con múltiples vocalizaciones, considerando que en este proyecto se realizó la identificación simultánea de especies utilizando audios individuales por especie, pero no se abordaron grabaciones con vocalizaciones mezcladas de varias especies. Además, el uso de hardware más avanzado, dado que el proyecto utilizó la GPU gratuita de Google Colab, podría mejorar significativamente los tiempos de entrenamiento y el rendimiento general del sistema.

Finalmente, este proyecto se distingue por integrar un modelo de aprendizaje profundo con una interfaz accesible para usuarios no técnicos, marcando un avance significativo en la clasificación bioacústica. Los resultados obtenidos no solo validan la efectividad de la metodología empleada, sino que también resaltan su contribución al estado del arte en este campo, estableciendo un punto de referencia para futuros estudios.

8. Conclusiones

- En relación con el primer objetivo, se logró ajustar el modelo ResNet-50 utilizando un conjunto de audios de las especies Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris. Este ajuste, realizado mediante la metodología CRISP-ML(Q) permitió abordar los desafíos de escasez de datos y variabilidad acústica, demostrando la efectividad de las estrategias de preprocesamiento y aumento de datos implementadas.
- Se evaluó el porcentaje global de acierto del modelo entrenado en la clasificación de vocalizaciones de Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris. El modelo alcanzó una precisión del 91% en el conjunto de entrenamiento y del 89.33% en el conjunto de prueba, lo que refleja un buen rendimiento tanto en los datos de entrenamiento como en los de prueba. Estos resultados confirman el cumplimiento del segundo objetivo, mostrando que el modelo ajustado puede clasificar de manera efectiva las vocalizaciones de las especies estudiadas.
- En la validación con datos externos, el modelo obtuvo un porcentaje global de acierto del 91.19%, lo que demuestra su capacidad para generalizar a nuevos datos. Este valor responde afirmativamente a la pregunta de investigación, confirmando que la transferencia de aprendizaje aplicada al modelo ResNet-50 permite lograr un alto porcentaje de acierto en la clasificación de vocalizaciones de las especies Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris.
- Se comprobó que el uso de espectrogramas como representación visual de los audios y la normalización de amplitud fueron factores determinantes para mejorar la calidad del conjunto de datos y el rendimiento del modelo, optimizando su capacidad para identificar patrones acústicos distintivos.
- La integración del modelo en una interfaz web permitió su evaluación en un entorno práctico, probándolo con audios externos mediante un enfoque tipo pipeline. Este sistema no solo demostró su funcionalidad, sino que también facilitó su accesibilidad para usuarios no especializados, promoviendo su aplicación en el monitoreo y conservación de especies.
- Este trabajo establece un marco sólido para la clasificación bioacústica en Ecuador, contribuye al desarrollo de herramientas tecnológicas que favorezcan la conservación de especies en regiones de alta diversidad biológica.

9. Recomendaciones

- Explorar estrategias para abordar la dependencia de un conjunto de datos reducido, incluyendo la recopilación de más grabaciones en diferentes hábitats, estaciones del año, lo que permitirá generar un dataset más robusto y representativo.
- Implementar técnicas avanzadas de eliminación de ruido y separación de fuentes para mejorar la calidad de los datos de entrada, especialmente en grabaciones con múltiples vocalizaciones o ruido ambiental significativo.
- Considerar el uso de hardware más avanzado, como GPUs dedicadas o servidores de alto rendimiento, para acelerar el proceso de entrenamiento y permitir la optimización de hiperparámetros de manera más eficiente.
- Ampliar la evaluación del modelo a especies adicionales, lo que contribuiría a validar su generalización y aplicabilidad en un mayor rango de contextos bioacústicos.
- Fomentar la colaboración interdisciplinaria con expertos en bioacústica, ornitología, ecología y tecnología para enriquecer el enfoque metodológico y explorar nuevas aplicaciones del modelo en proyectos de conservación.
- Aprovechar las tecnologías emergentes y modelos multimodales que surjan en el futuro, integrando datos acústicos, visuales y contextuales, con el objetivo de mejorar la precisión y versatilidad del sistema, incorporando los avances en inteligencia artificial para optimizar el rendimiento del proyecto.

10. Bibliografía

[1] S. Duan et al., "Acoustic component detection for automatic species recognition in environmental monitoring," *Ecological Informatics*, vol. 6, no. 6, pp. 381-391, Dec 2011.

[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)*, San Francisco, CA, USA, Jun 2016, pp. 770-778.

[3] W. Apolo and M. Becking, "Conservando el Parque Nacional Podocarpus mediante una Estrategia Corporativa de Desarrollo Sustentable," *Lyonia*, vol. 4, no. 2, pp. 109-114, 2003.

[4] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279-283, Mar 2017.

[5] C. Randier, "Comparing methods of instruction using bird species identification skills as indicators," *Biological Education*, vol. 36, no. 4, pp. 181-188, Dec 2002. DOI: 10.1080/00219266.2002.9655830.

[6] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification," *Research Square*, vol. 1, no. 1, pp. 1-17, Apr 2023. DOI: 10.21203/rs.3.rs-2863523/v1.

[7] S. Fagerlund, "Studies on Bird Vocalization Detection and Classification of Species," Aalto University, Espoo, Finlandia, Doctoral thesis ISBN 978-952-60-5922-8, 2014.

[8] B. Koonce, *ResNet 50 Convolutional Neural Networks with Swift for Tensorflow*, 1st ed., Jefferson, MO, Ed. USA: SpringerLink, 2021.

[9] S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, and R. C. Moore, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Los Angeles, CA, USA, 2017, pp. 1-5.

[10] J. Freile and F. Rodas, "Conservación de aves en Ecuador: ¿cómo estamos y qué necesitamos hacer?," *Cotinga*, vol. 29, no. 1, pp. 48-55, Dec 2008.

[11] F. Reyes, A. Orihuela, L. Ordóñez and D. Armijos, "Registros inusuales de aves en la hoya de Loja, Andes sur del Ecuador," *ACI Avances en Ciencias e Ingenierías*, vol. 8, no. 1, pp. 1-11, Feb 2016. DOI: 10.18272/aci.v8i14.276.

[12] R. Ridgely and G. Tudor, *The Birds of South America: The Suboscine Passerines*, 2nd ed. Texas, USA: University of Texas Press, 1994.

[13] N. Krabbe and T. Schulenberg, "A new species of Scytalopus tapaculo from the Ecuador-Peru border," *Wilson Bulletin*, vol. 111, no. 2, pp. 157-165, Jun 1999. DOI: 10.1676/12-055.1.

[14] David McDonald. (2024, Jan) "Macaulay Library", Cornell Lab of Ornithology. [Online]. https://macaulaylibrary.org/asset/613926876

[15] T. M. Donegan and B. C. Huertas, "Threatened species of the Serranía de los Yariguíes: Zoogeography and conservation of birds in the Andes of Colombia," *Pro aves*, vol. 4, pp. 39-77, Dec 2006.

[16] Lorenz Stephan. (2021, Aug) "Macaulay Library", Cornell Lab of Ornithology. [Online]. https://macaulaylibrary.org/asset/621187492

[17] J. V. Remsen et al., "A classification of the bird species of South America," *American Ornithologists" Union*, vol. 120, no. 2, pp. 565-575, Dec 2016.

[18] Andrés Vazquez Noboa. (2023, Oct) "Macaulay Library", Cornell Lab of Ornithology. [Online]. https://macaulaylibrary.org/asset/617178354

[19] V. Devictor, R. J. Whittaker, and C. Beltrame, "Beyond scarcity: Citizen science programmes as useful tools for conservation biogeography," *Diversity and Distributions*, vol. 16, no. 3, pp. 354-362, May 2010. DOI: 10.1111/j.1472-4642.2009.00615.x.

[20] Xeno-canto Foundation and Naturalis Biodiversity Center, "Xeno-canto: Sharing bird sounds from around the world". (2024, Nov) Xeno-canto.org. [Online]. https://xeno-canto.org/

[21] B. Sullivan et al., "eBird: A citizen-based bird observation network in the biological sciences," *Biological Conservation*, vol. 142, no. 10, pp. 2282-2292, Oct 2009. DOI: 10.1016/j.biocon.2009.05.006.

[22] W. Vellinga and R. Planqué, "The Xeno-canto collection: A global archive of bird sounds," *Biodiversity Data Journal*, vol. 3, no. e4701, pp. 1-10, Aug 2015. DOI: 10.3897/BDJ.3.e4701.

[23] S. Kahl, C. Wood, M. Eibl, and H. Klinck, "BirdNet: A deep learning solution for bird identification using audio recordings," *Ecological Informatics*, vol. 61, pp. 101-236, Mar 2021. DOI: 10.1016/j.ecoinf.2021.101236.

[24] S. Kahl et al., "Overview of BirdCLEF 2021: Bird call identification in soundscape recordings," in *CLEF 2021 Conference and Labs of the Evaluation Forum*, vol. 117, Bucharest, Romania, Feb 2021, pp. 1-12. DOI: 10.1007/978-3-030-85251-1_1.

[25] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2020. ISBN 978-0-13-461099-3.

[26] F. Chollet, *Deep Learning with Python*, 1st ed. Shelter Island, NJ, USA: Manning Publications, 2018. ISBN-10: 1617294438, ISBN-13: 978-1617294433.

[27] M. I. Jordan and T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospects," *Science*, vol. 349, no. 6245, pp. 255-260, Jul 2015. DOI: 10.1126/science.aaa8415.

[28] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, U.K.: The MIT Press, 2012. ISBN-10: 0262018020, ISBN-13: 978-0262018029.

[29] E. Alpaydin, *Introduction to Machine Learning*, 4th ed. Cambridge, U.K.: The MIT Press, 2020. ISBN-10: 0262043793, ISBN-13: 978-0262043793.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, U.K: The MIT Press, 2016. ISBN-10: 0262035618, ISBN-13: 978-0262035613.

[31] Y. Bengio, A. Courville, and P. Vincent, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1-127, Jan 2009. DOI: 10.1561/2200000006.

[32] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. Berlin, Germany: Springer, 2010. ISBN-10: 1848829345, ISBN-13: 978-1848829343.

[33] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Boston, MA, USA: Pearson, 2018. ISBN-10: 0133356728, ISBN-13: 978-0133356724.

[34] N. A. M. Zahid, M. S. Z. Ali, and H. M. Kamal, "An overview of image segmentation methods," *International Journal of Computer Applications*, vol. 116, no. 22, pp. 14-22, Apr 2015. DOI: 10.5120/20482-2850.

[35] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, Toronto, ON, Canada, 2012, pp. 1097-1105. DOI: 10.1145/3065386.

[36] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 4700-4708. DOI: 10.1109/CVPR.2017.243.

[37] 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, Nov 1998. DOI: 10.1109/5.726791.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016, pp. 630-645. DOI: 10.1007/978-3-319-46493-0_38.

[39] M. Rastegari, V. Ordonez, K. Redmond, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016, pp. 525-542. DOI: 10.1007/978-3-319-46493-0_32.

[40] S. Zhang, J. Li, and S. Bai, "ResNet-50 for Image Classification," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 123-135, Mar 2018. DOI: 10.1007/s11390-018-1816-5.

[41] SemilleroCV. (2024, Aug) "Hugging Face ResNet-50 Model". [Online]. https://huggingface.co/SemilleroCV/resnet50-finetuned-bwmp2-224

[42] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct 2010. DOI: 10.1109/TKDE.2009.191.

[43] J. Yosinski, A. Nguyen, J. Yang, and Y. LeCun, "How Transferable Are Features in Deep Neural Networks?," in *Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, USA, 2014, pp. 3320-3328. DOI: 10.5555/2969033.2969196.

[44] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019. ISBN-10: 1492032646, ISBN-13: 978-1492032649.

[45] J. Heaton, *Deep Learning and Neural Networks*, 1st ed. St. Louis, MO, USA: Heaton Research, Inc., 2018. ISBN-10: 1626344563, ISBN-13: 978-1626344568.

[46] J. Hahn and F. Pfenning, "CRISP-ML(Q): A Quality-Driven Extension of the CRISP-DM Process Model for Machine Learning," *Journal of Data Science*, vol. 17, no. 3, pp. 501-515, Sep 2019. DOI: 10.1007/s10618-019-00631-5.

[47] K. Kahn and Y. Matsuoka, "Measuring the quality of classification in machine learning: metrics and methods," *Journal of Computational and Graphical Statistics*, vol. 26, no. 4, pp. 927-946, Dec 2017. DOI: 10.1080/10618600.2017.1366912.

[48] J. H. Friedman, "On bias, variance, 0/1 loss, and the curse of dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55-77, Mar 1997. DOI: 10.1023/A:1009778005914.

[49] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks," *Journal of Machine Learning Research*, vol. 7, pp. 1865-1894, Jul 2006.

[50] D. M. Powers, "Evaluation: From Precision, Recall and F-Score to ROC, AUC, FDR and Back Again," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37-63, Jan 2016. DOI: 10.15346/hc.v2i1.1.

[51] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1-13, 2020. DOI: 10.1186/s12864-019-6413-7.

[52] J. D. Johnston, "Transform Coding of Audio Signals," *IEEE Journal on Selected Areas in Communications*, vol. 3, no. 5, pp. 820-825, Sep 1985. DOI: 10.1109/JSAC.1985.1146231.

[53] P. L. Bartlett, H. Y. Cheang, S. M. Drummond, and F. A. Rincon, "The Impact of Bit Depth on Audio Quality," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 517-529, Mar 2016. DOI: 10.1109/TASLP.2016.2522421.

[54] K. W. Cheung, C. T. Hsiao, and J. M. MacDonald, "Sample Rate Conversion: Theory and Applications," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 6, pp. 1800-1809, Nov 2007. DOI: 10.1109/TASL.2007.905178.

[55] H. T. Tanaka, A. J. Eastwood, and H. S. Kimura, "Implications of Sampling Rate in Digital Audio Systems," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 48-55, Dec 2016. DOI: 10.1109/MCOM.2016.1600378.

[56] J. Watkinson, *The Art of Digital Audio Recording: A Practical Guide for Home and Studio*, 1st ed. Oxford, England: Oxford University Press, 2007. ISBN-10: 0240520519, ISBN-13: 978-0240520512.

[57] R. A. Morrison, "The MP3 Revolution: How MP3 and Other Digital Formats Have Changed Music," *Music Technology Journal*, vol. 10, no. 1, pp. 20-30, Jan 2017.

[58] X. Zhang and J. Hu, "An Overview of Audio File Formats: MP3, WAV, and Beyond," *Journal of Multimedia*, vol. 8, no. 3, pp. 430-440, May 2016. DOI: 10.4304/jmm.8.3.430-440.

[59] J. R. Bradley, "Understanding Digital Audio Formats: MP3 vs. WAV," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 5, pp. 883-895, May 2018. DOI: 10.1109/TASLP.2018.2819987.

[60] R. Mitchell, *Web Scraping with Python: Collecting More Data from the Modern Web*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018. ISBN-10: 1491985543, ISBN-13: 978-1491985544.

[61] S. Cross and B. Carver, "Creative Commons: The Next Generation of Copyright Licenses," *Journal of Intellectual Property Law & Practice*, vol. 12, no. 4, pp. 281-290, Apr 2017. DOI: 10.1093/jiplp/jpx001.

[62] Z. Katz, "Creative Commons Licenses: What Every User Should Know," *Journal of Digital Media Law*, vol. 5, no. 2, pp. 90-105, Jun 2016.

[63] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*, 1st ed. Chichester, England: Wiley, 2010. ISBN-10: 0470997658, ISBN-13: 978-0470997659.

[64] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2006. ISBN-10: 0131873741, ISBN-13: 978-0131873742.

[65] R. W. Schafer, M. A. Yoder, and J. H. McClellan, *DSP First: A Multimedia Approach*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall, 2003. ISBN-10: 0130909998, ISBN-13: 978-0130909992.

[66] D. Fitzgerald, *Audio Signal Processing and Coding*, 1st ed. Hoboken, NJ, USA: John Wiley & Sons, 2009. ISBN-10: 0470041966, ISBN-13: 978-0470041964.

[67] K. S. Rajasekaran and R. R. Dhanalakshmi, *Audio Signal Processing: Principles and Applications*, 1st ed. Singapore: Springer, 2017. ISBN-10: 9811027643, ISBN-13: 978-9811027645.

[68] J. O. Smith, *Spectral Audio Signal Processing*, 1st ed. Stanford, CA, USA: W3K Publishing, 2007. ISBN-10: 0974560728, ISBN-13: 978-0974560727.

[69] R. W. Rabiner and L. R. Schafer, "Digital Processing of Speech Signals," *IEEE Proceedings*, vol. 63, no. 4, pp. 560-581, Apr 1975. DOI: 10.1109/PROC.1975.9792.

[70] H. Asif, "Wavelet Transform Based Audio Signal Processing," *IEEE Access*, vol. 8, pp. 23572-23587, 2020.

[71] D. Wang and G. Hu, "A study of the Effects of Noise on Audio Signal Processing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 290-298, Feb 2010.

[72] J. Choi, "Normalization Techniques in Digital Signal Processing," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 3972-3984, Aug 2012. DOI: 10.1109/TSP.2012.2200887.

[73] G. T. Dai, "Data Augmentation Techniques for Deep Learning in Medical Imaging: A Review," *IEEE Reviews in Biomedical Engineering*, vol. 13, pp. 159-173, 2020. DOI: 10.1109/RBME.2019.2957171.

[74] H. M. Lee, "Improving the Generalization of Speech Recognition Systems Using Data Augmentation Techniques," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 3, pp. 517-528, Mar 2019. DOI: 10.1109/TASLP.2019.2894967.

[75] D. George, "Combining Data Augmentation and Ensemble Learning for Enhanced Audio Classification," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 497-501, Apr 2018. DOI: 10.1109/LSP.2018.2803045.

[76] J. Kim, "Optimizing Deep Learning Models: An Overview," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 1155-1169, Apr 2019. DOI: 10.1109/TNNLS.2018.2876178.

[77] M. Hinton, "Understanding Regularization Techniques for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1035-1052, Mar 2021. DOI: 10.1109/TNNLS.2020.2996184.

[78] S. Gugger, "FastAI: A Practical Approach to Deep Learning," *IEEE Software*, vol. 37, no. 4, pp. 48-55, Jul 2020.

[79] S. Rajan, "Leveraging FastAI for Effective Model Training and Evaluation," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 300-308, Jan 2021.

[80] B. Jones and D. Beazley, *Python Cookbook: Recipes for Mastering Python 3*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2013. ISBN-10: 1449340377, ISBN-13: 978-1449340377.

[81] T. E. Oliphant, "Python for Scientific Computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10-20, May 2007. DOI: 10.1109/MCSE.2007.58.

[82] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017. ISBN-10: 1491957662, ISBN-13: 978-1491957660.

[83] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference (SciPy)*, Austin, TX, USA, 2010, pp. 51-56. DOI: 10.25080/Majora-92bf1922-00a.

[84] T. Oliphant and G. van Rossum. (2017) "Requests: HTTP for Humans". [Online]. https://requests.readthedocs.io/en/latest/

[85] L. Costa. (2020) "tqdm Documentation". [Online]. https://github.com/tqdm/tqdm

[86] J. Robertson. (2020) "Pydub Documentation: Manipulating Audio with Python". [Online]. https://pydub.com

[87] B. McFee et al., "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference (SciPy)*, Austin, TX, USA, 2015, pp. 18-25. DOI: 10.25080/Majora-7b98e3ed-003.

[88] P. Daniel and B. McFee, "Analyzing Song Structure with a Timbre-Based Representation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 2, pp. 512-519, Feb 2014. DOI: 10.1109/TASLP.2013.2296200.

[89] C. Arnaud et al., "torchaudio: Building Blocks for Audio and Speech Processing with PyTorch," in *Proceedings of the 2020 International Conference on Machine Learning (ICML)*, New York, NY, USA, 2020, pp. 50-70. DOI: 10.48550/arXiv.2110.15018.

[90] P. Mermelstein, "Distance Measures for Speech Recognition, Psychological and Instrumental," in *Pattern Recognition and Artificial Intelligence*, Montreal, Canada, 1976, pp. 374-388.

[91] J. Howard and S. Gugger, *Deep Learning for Coders with Fastai and PyTorch: AI Applications Without a PhD*, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2020. ISBN-10: 1492045527, ISBN-13: 978-1492045526.

[92] M. Sutskever and S. Krizhevsky, "An Empirical Study on the DataBlock API of FastAI," *IEEE Transactions on Learning Technologies*, vol. 15, no. 2, pp. 129-140, Jun 2022. DOI: 10.3390/info11020108.

[93] E. Bisong, "Google Colaboratory," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Berkeley, CA, USA: Apress, 2020, pp. 59-64. ISBN-10: 1484257651, ISBN-13: 978-1484257652.

[94] B. Chandu, A. Munikoti, K. Murthy, G. Murthy, and C. Nagaraj, "Automated bird species identification using audio signal processing and neural networks," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, 2020, pp. 1-5. DOI: 10.1109/AISP48273.2020.9073584.

[95] N. Priyadarshani, S. Marsland, and I. Castro, "Automated birdsong recognition in complex acoustic environments: a review," *Journal of Avian Biology*, vol. 49, no. 5, pp. jav-01447, 2018. DOI: 10.1111/jav.01447.

[96] S. Rassak, M. Nachamai, and A. Krishna, "Survey study on the methods of bird vocalization classification," in *2016 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 2016, pp. 1-8. DOI: 10.1109/ICCTAC.2016.7567337.

[97] S. A. Al-Qbailat and S. T. Al-Showarah, "Birds identification system using deep learning," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 4, 2021. DOI: 10.14569/IJACSA.2021.0120434.

[98] M. Lasseck, "Bird Species Identification in Soundscapes," *CLEF (Working Notes)*, vol. 2380, 2019.

[99] S. Disabato, G. Canonaco, P. Flikkema, M. Roveri, and C. Alippi, "Birdsong detection at the edge with deep learning," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2021, pp. 9-16. DOI: 10.1109/SMARTCOMP52413.2021.00022.

[100] V. S. Prasad, "Bird Species Identification Through Vocalization Analysis Using Machine Learning," *International Journal of Information Technology and Computer Engineering*, vol. 12, no. 3, pp. 742-749, 2024.

[101] P. Jancovic and M. Kokuer, "Bird species recognition using unsupervised modeling of individual vocalization elements," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 5, pp. 932-947, 2019. DOI: 10.1109/TASLP.2019.2904790.

[102] M. Sankupellay and D. Konovalov, "Bird call recognition using deep convolutional neural network, ResNet-50," *Proc. Acoustics*, vol. 7, pp. 1-8, 2018.

[103] D. Efremova, M. Sankupellay, and D. Konovalov, "Data-efficient classification of birdcall through convolutional neural networks transfer learning," in *2019 Digital image computing: Techniques and applications (DICTA)*, 2019, pp. 1-8. DOI: 10.48550/arXiv.1909.07526.

[104] PARKS & TRIBES. (2024, Noviembre) Información sobre localización y servicios, Agencia de Viajes en Quito. [Online]. https://www.parks-and-tribes.com/national-parks/parque-nacional-podocarpus/parque-nacional-podocarpus.htm

11. Anexos

Anexo 1: Entrevista al Ingeniero Oscar Cumbicus
https://drive.google.com/file/d/1aBcQOjxrPps6kZONKdFG4w0y--fABmTs/view?usp=sharing

Anexo 2: Código del proyecto (Proceso de creación del modelo)
https://colab.research.google.com/drive/1G29kh3EF3OQg_us9Swtb4qcmWpBx0c7P?usp=drive_link

Anexo 3: Especificaciones técnicas del modelo empaquetado

En la **Tabla 15**, se presentan las especificaciones más importantes del modelo empaquetado para su uso.

Tabla 15: Especificaciones técnicas del modelo empaquetado

Parámetro	Valor
Arquitectura	ResNet-50
Número de clases	3
Función de pérdida	CrossEntropyLossFlat
Optimizador	Adam
Tasa de aprendizaje	10^{-3}

Diferencia entre formatos .pth y .pkl

- **.pth:** Archivo de PyTorch que contiene solo los pesos del modelo, ideal para reentrenar o continuar con ajustes.
- **.pkl:** Archivo empaquetado que incluye pesos y la configuración del modelo, perfecto para la implementación directa en producción.

Enlace modelo en formato (.pth):
https://drive.google.com/file/d/1-0B2Kx1HPSewlfZm5LWFUS6YTk-M31l3/view?usp=drive_link

Enlace modelo en formato (.pkl):
https://drive.google.com/file/d/1R6RcnE5j7SQatEcIckCujmtV3yBXakRp/view?usp=drive_link

Anexo 4: Ejecución de la aplicación web paso a paso

Paso 1: Selección del archivo de audio

El proceso comienza con la selección de un archivo de audio desde el administrador de archivos. Al hacer clic en el botón “Seleccionar Archivo”, se abre una ventana para buscar y cargar el archivo deseado. En la **Figura 47**, en este ejemplo, se seleccionó el archivo Audio5.mp3.

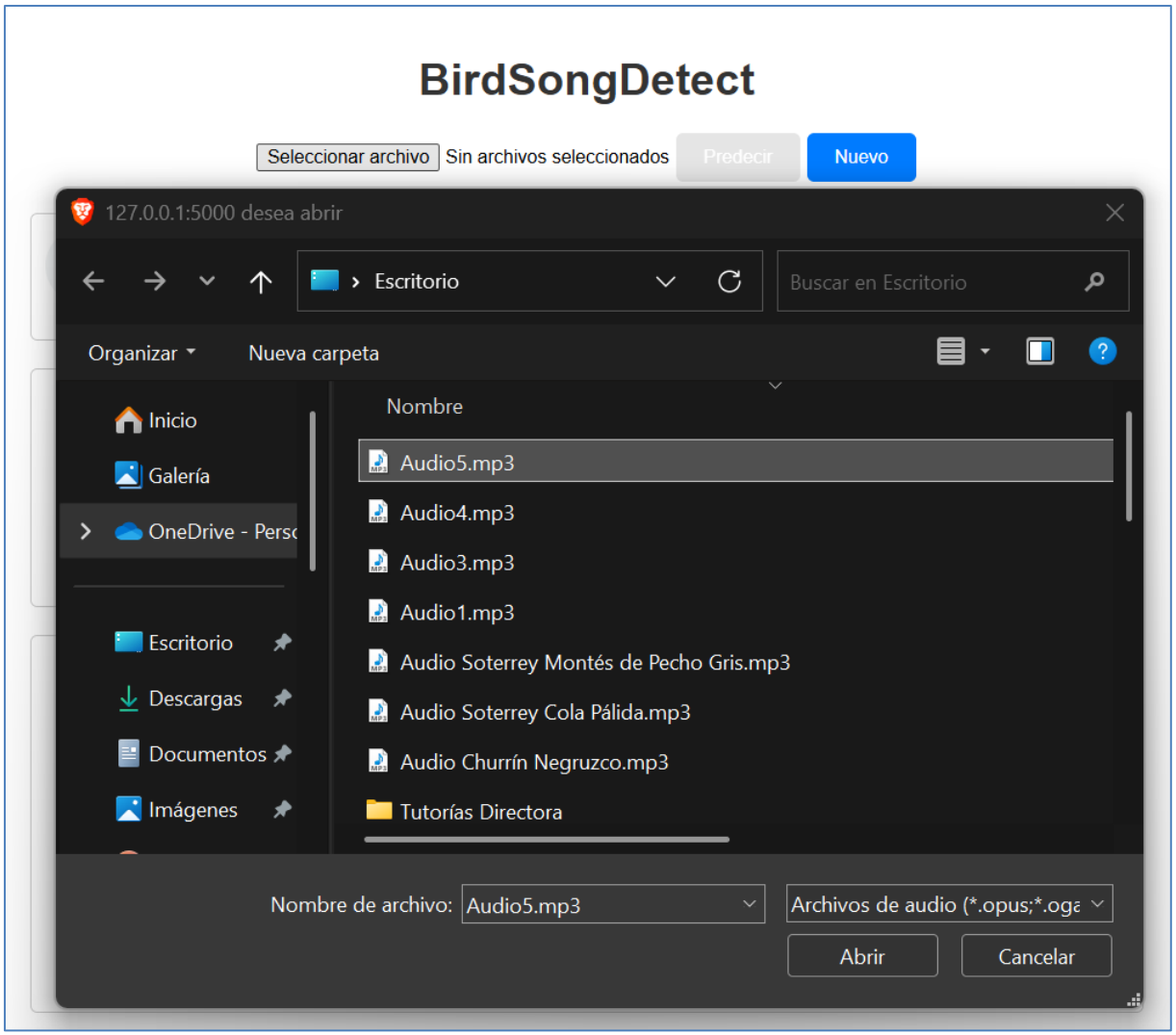


Figura 47: Interfaz Web - Paso 1: Selección del archivo de audio

Paso 2: Predicción de la especie

Una vez cargado el archivo, el botón "Predecir" y el reproductor se habilitan. Al hacer clic en este botón, el modelo empaquetado procesa el audio y realiza la predicción. En la **Figura 48**, durante este proceso, se muestra una barra de progreso indicando que el modelo está trabajando.

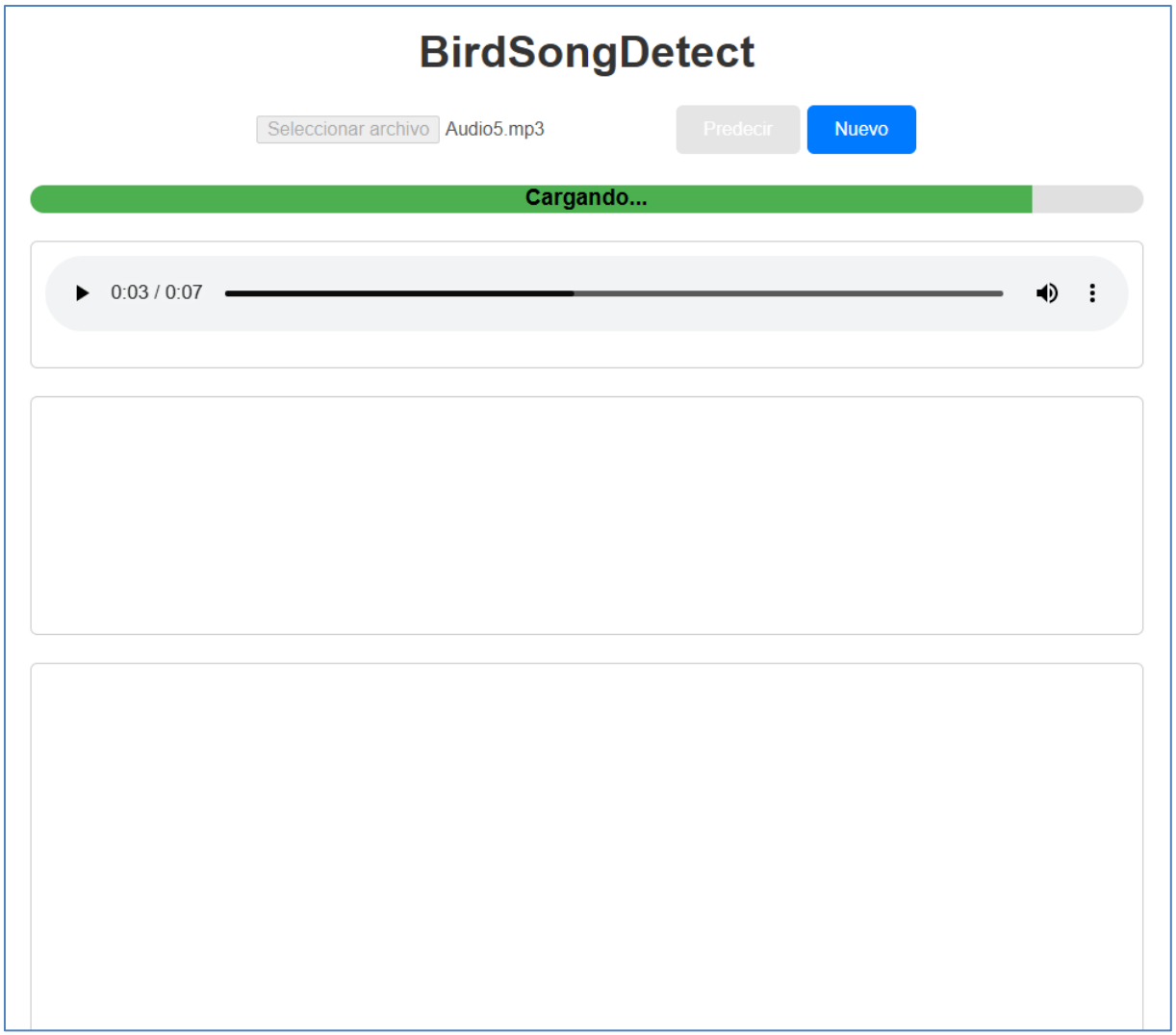


Figura 48: Interfaz Web - Paso 2: Predicción de la especie

Paso 3: Visualización de la predicción

Tras completar la predicción, la interfaz muestra los resultados. Estos incluyen la forma de onda y el espectrograma que son representaciones gráficas generadas a partir del audio. Además, se presenta el nombre de la especie acompañado de una imagen propia. En la **Figura 49**, el modelo predijo correctamente la especie **Churrín Negruzco**.

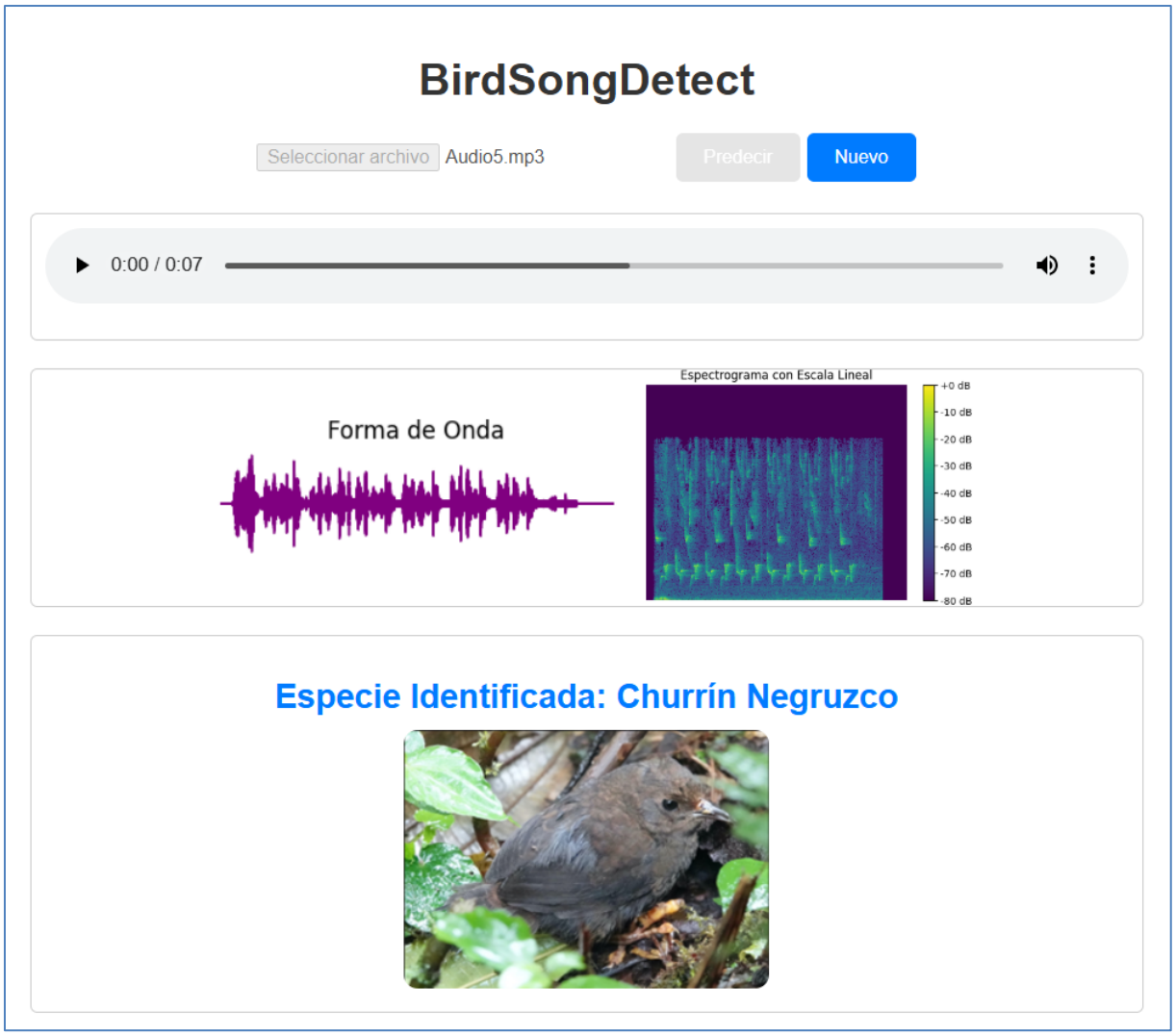


Figura 49: Interfaz Web - Paso 3: Visualización de la predicción

Enlace de un video explicativo sobre el uso del sitio web:
https://drive.google.com/file/d/1q45azFfRb665eqm6nysEWNF35w3oJzEN/view?usp=drive_link

Enlace del proyecto hecho en Flask:
https://drive.google.com/file/d/1iBjFkUM7lcwi1BpFTEG1DW-M5Ga_5emA/view?usp=drive_link

Anexo 5: Certificado de traducción del resumen por parte de un profesional

UNIVERSIDAD NACIONAL DE LOJA
ÁREA DE LAS ENERGÍAS, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES
COMPUTACIÓN

Loja, 03 de febrero de 2025

Lic. Stefanny Susana Cajamarca Bustamante
LICENCIADO EN CIENCIAS DE LA EDUCACIÓN MENCIÓN INGLÉS

CERTIFICO:

Que el resumen del Trabajo de Integración Curricular (TIC) titulado ***"Transferencia de Aprendizaje Hacia el Modelo ResNet-50 para la Clasificación de Especies de Aves como Churrín Negruzco, Soterrey Cola Pálida y Soterrey Montés de Pecho Gris"***, elaborado por el estudiante Jimmy Alexander Cajamarca Escaleras, portador de la cédula de identidad Nro. **1106070889**, de la Carrera de Computación de la Universidad Nacional de Loja, ha sido traducido fielmente al idioma inglés, manteniendo la coherencia, precisión y estructura propias de la lengua extranjera.

Lo certifico en honor a la verdad y autorizo el uso del presente documento conforme a los intereses del solicitante.

Lic. Stefanny Susana Cajamarca Bustamante
CI: 1105030801
Nro. Reg. Senescyt: 1031-2017-1878623
LICENCIADO EN CIENCIAS DE LA EDUCACIÓN MENCIÓN INGLÉS