PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

SISTEMAS DE INFORMACIÓN

DESARROLLO DE UN SISTEMA DE APRENDIZAJE AUTOMÁTICO PARA LA CLASIFICACIÓN DE AVES DE LA REGIÓN ANDINA DEL ECUADOR

AUTOR:

ISAAC DARÍO CUEVA CRIOLLO

QUITO DM, JULIO DE 2025

DEDICATORIA

A Dios y a mi familia, por ser testigos de cada paso en este proceso de superación y crecimiento personal. Ellos me enseñaron que detrás de un gran profesional siempre habita una excelente persona. Cada logro alcanzado y cada meta cumplida son reflejo de su amor, apoyo incondicional y confianza en mí. Todos mis éxitos les pertenecen por completo.

AGRADECIMIENTO

A mis padres, por su apoyo y amor incondicional. Gracias por enseñarme que el trabajo duro, honesto y disciplinado es la clave para triunfar en la vida.

Al Ing. Charles Escobar, por compartir su conocimiento en el campo de la Ciencia de Datos y su apoyo en la realización de este proyecto.

A todos aquellos que me brindaron su apoyo y me acompañaron a lo largo de este proceso de formación profesional. Su amistad y confianza representan un recuerdo invaluable que llevaré conmigo siempre.

RESUMEN

El presente trabajo tiene como objetivo desarrollar un sistema de aprendizaje automático capaz de clasificar imágenes de aves de la región andina del Ecuador, contribuyendo a la divulgación del conocimiento sobre las especies locales mediante una herramienta tecnológica intuitiva. Para alcanzar este propósito, se construyó un modelo basado en redes neuronales convolucionales utilizando TensorFlow y Keras, entrenado con un conjunto de datos recopilado de repositorios variados y sometido a un proceso de selección, preprocesamiento y ajuste. Además, se incorporó una capa de visualización desarrollada en Django, que permite al usuario interactuar con el sistema y consultar de manera clara las predicciones generadas por el modelo. Los resultados obtenidos evidencian que el modelo entrenado con imágenes en formato RGB alcanzó una precisión global superior al 80% en la clasificación multiclase de cinco especies de aves representativas, confirmando que la información cromática constituye un factor determinante para mejorar la capacidad de discriminación visual en entornos naturales con alta diversidad de colores y formas.

Palabras clave: aprendizaje profundo, redes neuronales convolucionales, TensorFlow, clasificación automática, aves andinas.

ABSTRACT

The present work aims to develop a machine learning system capable of classifying images of birds from the Andean region of Ecuador, contributing to the dissemination of knowledge about local species through an intuitive technological tool. To achieve this objective, a model based on convolutional neural networks was built using TensorFlow and Keras, trained with a dataset collected from various repositories and subjected to a process of selection, preprocessing, and adjustment. In addition, a visualization layer developed with Django was incorporated, allowing users to interact with the system and clearly consult the predictions generated by the model. The results show that the model trained with RGB images achieved an overall accuracy exceeding 80% in the multiclass classification of five representative bird species, confirming that chromatic information is a determining factor in improving visual discrimination capacity in natural environments with high diversity of colors and shapes.

Keywords: deep learning, convolutional neural networks, TensorFlow, automatic classification, Andean birds.

ÍNDICE DE CONTENIDO

CAPÍTULO I – INTRODUCCIÓN 12

1.1 Justificación 12

1.2 Planteamiento del Problema 12

1.3 Objetivos 15

1.3.1 Objetivo General 15

1.3.2 Objetivos Específicos 15

1.4 Antecedentes 15

1.6 Alcance 17

CAPÍTULO II – MARCO TEÓRICO 18

2.1. Inteligencia Artificial 18

2.2 Aprendizaje Automático y Deep Learning 19

2.3 Redes Neuronales 20

2.3.1 Redes Neuronales Convolucionales 21

2.4 Representación Cromática 22

2.5 Metodología CRISP-DM 23

2.6. Métricas de Evaluación de Modelos de Clasificación 24

2.7 Tecnologías y Herramientas 25

2.7.1. TensorFlow y Keras 25

2.7.2. Figma 27

2.7.3. Django 27

CAPÍTULO III – METODOLOGÍA 29

3.1 Investigación Científica 29
3.1.1 Enfoque y Tipo de Investigación 29
3.1.2 Población y Muestra 30
3.2 Investigación Técnica 33
3.2.1 Enfoque Metodológico CRISP-DM 33
CAPÍTULO IV – DESARROLLO DEL SISTEMA DE APRENDIZAJE AUTOMÁTICO 37
4.1 Organización y Preparación del Conjunto de Datos de Aves 37
4.1.1 Entendimiento del Negocio 37
4.1.2 Entendimiento de los Datos 38
4.1.2.1 Extracción de Datos API iNaturalist 39
4.1.2.2 Análisis Exploratorio y Estadístico de Imágenes. 43
4.1.2.3 Implicaciones del Análisis para la Calidad de las Imágenes. 50
4.1.3 Preparación de los Datos 51
4.1.3.1 Filtrado y Selección de Imágenes Relevantes. 51
4.1.3.2 Estandarización de Atributos de Imagen. 54
4.1.3.3 Data Augmentation. 58
4.2 Construcción del Modelo de Clasificación 59
4.2.1 Modelado 59
4.2.1.1 Arquitectura Red Neuronal Convolucional – RGB. 60
4.2.1.2 Arquitectura Red Neuronal Convolucional – Escala de Grises. 66
4.3 Evaluación del Desempeño del Modelo de Clasificación 68
4.3.1 Evaluación 69
4.3.1.1 Visualización del Rendimiento del Modelo – RGB. 69

4.3.1.2 Visualización del Rendimiento del Modelo – Escala de Grises. ...71
4.3.1.3 Demostración de la Capacidad Predictiva del Modelo. ...73
4.3.1.4 Análisis de las Métricas de Evaluación. ...75
4.4 Capa de Visualización e Interacción del Modelo ...82
4.4.1 Despliegue ...83
4.4.1.1 Arquitectura y Diseño de la Interfaz. ...83
4.4.1.2 Integración de Datos y Visualización Final. ...86
CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES ...92
5.1 Conclusiones ...92
5.2 Recomendaciones ...94
BIBLIOGRAFÍA ...96
ANEXOS ...99

ÍNDICE DE FIGURAS

Figura 1: Inteligencia Artificial, Machine Learning y Deep Learning (IASolver, s.f.) 20

Figura 2: Funcionamiento de una Red Neuronal Convolucional (LIS Data Solutions, s.f.) 22

Figura 3: Ciclo de vida metodología CRISP-DM (IBM, 2020) 24

Figura 4: Funcionamiento MTV de Django (EspiFreelancer, s.f.) 28

Figura 5: Clarinero Escarlata (Anisognathus igniventris) (© Nicole LaRoche) 30

Figura 6: Carpintero Candela Común (Colaptes rivolii) (© Karl Kroeker) 31

Figura 7: Águila Pechinegra (Geranoaetus melanoleucus) (© Sebastián Lescano) 31

Figura 8: Pava Andina (Penelope montagnii) (© cmmteixeira) 32

Figura 9: Cóndor Andino (Vultur gryphus) (© Luis Enrique Calderón Franco) 32

Figura 10: Ejecución del script para la descarga masiva de imágenes desde iNaturalist 40

Figura 11: Función para la extracción de imágenes y metadatos desde la API iNaturalist 42

Figura 12: Gráfico de barras de la distribución de imágenes por especie 44

Figura 13: Gráfico de barras de las dimensiones promedio (ancho y alto) por especie 45

Figura 14: Gráfico de barras de brillo y contraste promedio por especie 47

Figura 15: Histogramas de intensidad de píxeles para el conjunto Cóndor Andino 48

Figura 16: Distribución de la variabilidad de píxeles por especie 49

Figura 17: Interfaz interactiva para la curación manual de imágenes 52

Figura 18: Implementación del proceso de verificación de archivos de imagen 53

Figura 19: Implementación de la conversión de modos de color a RGB 54

Figura 20: Implementación del proceso de redimensionamiento de imágenes 55

Figura 21: Muestra de imágenes estandarizadas del Clarinero Escarlata 55

Figura 22: Muestra de imágenes de las cinco especies normalizadas.....56
Figura 23: Comparación visual de imagen original y con filtro Gaussiano aplicado 57
Figura 24: Muestra de imágenes generadas por aumento de datos del conjunto Pava Andina 59
Figura 25: Configuración de ImageDataGenerator para la división de datos 64
Figura 26: Configuración y Ejecución del Entrenamiento del Modelo CNN 65
Figura 27: Muestra de imágenes transformadas a escala de grises para cada especie 66
Figura 28: Detalle de la época final del Entrenamiento del Modelo CNN-Escala de Grises.....68
Figura 29: Curvas de Precisión del Modelo CNN-RGB durante el Entrenamiento 70
Figura 30: Curvas de Pérdida del Modelo CNN-RGB durante el Entrenamiento.....71
Figura 31: Curvas de Precisión del Modelo CNN-Escala de Grises durante el Entrenamiento....72
Figura 32: Curvas de Pérdida del Modelo CNN-Escala de Grises durante el Entrenamiento 73
Figura 33: Predicciones del Modelo CNN-RGB sobre imágenes nuevas.....74
Figura 34: Predicciones del Modelo CNN-Escala de Grises sobre imágenes nuevas 75
Figura 35: Matriz de Confusión del Modelo RGB 78
Figura 36: Matriz de Confusión del Modelo Escala de Grises.....82
Figura 37: Diagrama de la Arquitectura MVT del Sistema 84
Figura 38: Prototipo de Interfaz de Usuario Diseñado en Figma 85
Figura 39: Carga del modelo CNN entrenado en Django 86
Figura 40: Panel de administración de Django con las especies registradas.....87
Figura 41: Generación del contexto y renderizado de la respuesta 88
Figura 42: Pantalla de Inicio del Sistema KUNTUR ML 89
Figura 43: Pantalla del Calsificador del Sistema KUNTUR ML 90
Figura 44: Pantalla Informativa del Sistema KUNTUR ML.....91

ÍNDICE DE TABLAS

Tabla 1: Especies seleccionadas para la descarga de observaciones desde iNaturalist 40

Tabla 2: Distribución de formatos de imagen por especie 44

Tabla 3: Distribución de modos de color de imágenes por especie 46

Tabla 4: Arquitectura y Resumen de Capas del Modelo CNN-RGB 62

Tabla 5: Arquitectura y Resumen de Capas del Modelo CNN-Escala de Grises 67

Tabla 6: Métricas de Evaluación para el Modelo CNN-RGB 76

Tabla 7: Métricas de Evaluación para el Modelo CNN-Escala de Grises 80

CAPÍTULO I – INTRODUCCIÓN

1.1 Justificación

Ecuador es un país rico en biodiversidad, especialmente en la Región Andina, donde la invaluable cantidad de especies endémicas lo hacen un patrimonio natural que genera interés de aprendizaje entre la comunidad. De tal forma que, tener una herramienta de clasificación precisa sobre las especies de esta área se convierte en una oportunidad de divulgación científica, educación ambiental e iniciativas de conservación y preservación.

En los últimos años el campo del reconocimiento de imágenes ha avanzado significativamente, donde el uso de algoritmos de clasificación basados en visión por computadora ofrece una oportunidad única para desarrollar un sistema de aprendizaje automático, el cual se encargue de clasificar automáticamente las aves a partir de fotografías. Impulsando el acceso a la información sobre aves locales y permitiendo a expertos, estudiantes, turistas y la comunidad en general identificar las especies con mayor facilidad y precisión. Su potencial impacto también se fundamenta en la capacidad de integración con otros sistemas que, en conjunto promuevan la conservación participativa, el ecoturismo y la educación ambiental en Ecuador.

1.2 Planteamiento del Problema

La riqueza natural de fauna de aves en la Región Andina del Ecuador siempre ha sido motivo de interés en la comunidad educativa y científica por su potencial uso para documentar actividad, estudio de comportamiento, medidas de conservación o simplemente observación y conocimiento. Esta amplia variedad ha generado gran cantidad de contenido fotográfico disperso por diversos medios y fuentes que, si bien es útil y de alto valor, en su gran mayoría no está

clasificada adecuadamente y presenta incoherencias por el gran número de similitudes que existen entre especies. De tal forma que el acceso a la información clara, precisa, actualizada y sobretodo correctamente catalogada sigue siendo un problema para el público en general.

Investigadores, estudiantes, turistas, guías naturalistas o ciudadanos con interés en la fauna suelen enfrentar dificultades para identificar las especies de aves que encuentran, especialmente si no cuentan con conocimientos especializados en biología o herramientas adecuadas para su reconocimiento. Esta barrera no solo limita el aprendizaje y la divulgación científica, sino que también reduce las posibilidades de participación activa en procesos de monitoreo o acciones de conservación y preservación.

En este contexto, no se ha desarrollado una herramienta tecnológica que permita automatizar y facilitar el reconocimiento y la clasificación de especies a partir de recursos visuales como fotografías, que hoy en día representan el medio más accesible y común de recolección de información por la facilidad de contar con un dispositivo móvil. Además, esta problemática no se reduce únicamente a la falta de una solución tecnológica, sino que se ve agravada por el grado de dificultad que implica desarrollar un modelo de aprendizaje automático con los recursos visuales disponibles de diversos repositorios, por lo cual, es necesario aplicar varios enfoques técnicos que permitan solventar los principales problemas a la hora de procesar imágenes. Entre los principales se encuentran: la variabilidad en la calidad de las imágenes, las condiciones ambientales en que se toman las fotos, la similitud visual entre algunas especies y todo el preprocesamiento requerido para alcanzar un grado de clasificación adecuado en el modelo.

A su vez, es necesario también pensar en la facilidad de uso para el usuario final, el sistema de clasificación requiere de una interfaz gráfica sencilla que integre el modelo y pueda otorgar la información correcta, por lo cual, las métricas de evaluación indicarán si es necesario buscar correcciones a fin de entregarle al usuario una clasificación acertada. Por tanto, resulta imprescindible reflexionar sobre las implicaciones técnicas, así como los elementos que dificultan la disponibilidad de un recurso tecnológico sólido y eficaz que permita a las personas no solo identificar, conocer y valorar la avifauna de la región, sino que también lo pueda integran en su entorno educativo o científico.

De la siguiente problematización, se despliega el siguiente problema principal:

- ¿Cómo se puede desarrollar un sistema de aprendizaje automático que permita la clasificación de aves a partir de imágenes de forma precisa y eficiente?

Y los siguientes problemas principales:

- ¿Qué características visuales y técnicas de preprocesamiento de imágenes son más relevantes para asegurar un rendimiento óptimo del modelo de clasificación de aves?
- ¿Qué enfoques en la construcción y procesamiento del modelo son más efectivos para clasificar adecuadamente las especies de aves?
- ¿Qué métricas de evaluación son las más óptimas para medir con precisión el rendimiento del modelo de clasificación de aves?
- ¿Cómo se puede implementar una capa de visualización que permita a los usuarios interactuar de forma sencilla con el modelo de clasificación y visualizar sus resultados de manera comprensible?

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema de aprendizaje automático capaz de clasificar imágenes de aves de la Región Andina del Ecuador.

1.3.2 Objetivos Específicos

- Organizar un conjunto de datos de imágenes de aves representativas de la Región Andina del Ecuador para el entrenamiento del modelo de clasificación.
- Construir un modelo de clasificación de imágenes basado en distintos enfoques de aprendizaje automático capaz de identificar y clasificar especies de aves.
- Evaluar el desempeño del modelo de clasificación a través de métricas apropiadas para determinar su desempeño.
- Implementar una capa de visualización que permita al usuario la interacción con el modelo.

1.4 Antecedentes

En los últimos años, el campo del aprendizaje profundo y visión por computadora ha experimentado avances significativos en la clasificación automática de imágenes, particularmente en aplicaciones biológicas muy conocidas que permiten identificar cualquier especie con tan solo una foto desde cualquier ángulo. Para lograrlo se ha utilizado una técnica llamada redes neuronales convolucionales (CNN), la cual ha demostrado ser una de las metodologías más efectivas en tareas de clasificación de imágenes debido a su capacidad para aprender de manera jerárquica las características visuales en las imágenes. Un ejemplo famoso es el caso de “Merlin Bird ID” de la Universidad de Cornell, una aplicación pionera en el uso de esta tecnología para el reconocimiento de aves, utilizando modelos entrenados con grandes bases de datos fotográficas para ayudar tanto

a expertos como a aficionados en la identificación de aves. Específicamente su función “Photo ID” permite clasificar aves a través de fotografías desde cualquier postura y ángulo. Su éxito radica en el amplio repositorio de Macaulay Library, donde se guardan miles de fotos y a medida que aumente, la aplicación mejorara exponencialmente (Cornell Lab of Ornithology, s.f.).

Asimismo, bibliotecas de aprendizaje profundo como TensorFlow y PyTorch, han permitido que los modelos de clasificación de imágenes se implementen de manera más eficiente y con mayor precisión en entornos definidos. Estas herramientas han abierto nuevas oportunidades para el desarrollo de modelos específicos para el reconocimiento de aves, no solo limitándose a imágenes sino también al procesamiento de audio, como lo demuestra el trabajo de Hoyos et al. (2021) en Colombia, quienes implementaron una técnica automatizada para la detección de vocalizaciones de aves mediante el análisis de frecuencias y aprendizaje automático. Utilizando más de 6,000 grabaciones de aves en el embalse Jaguas, lograron identificar vocalizaciones específicas, destacando el potencial de los modelos de aprendizaje automático en la clasificación de especies en condiciones acústicas diversas.

En el ámbito local la Universidad de Guayaquil presentó un trabajo atribuido a Quimis Barzola & Ramírez Reyes (2019), el cual desarrollo un sistema de clasificación de aves mediante un comedero automático con cámaras y sensores, utilizando redes neuronales convolucionales (CNN) para interpretar imágenes y reconocer especies. A pesar de sus esfuerzos, los resultados no fueron óptimos debido a limitaciones técnicas del hardware utilizado, lo que afectó directamente en la calidad de las fotografías y su eficiencia al momento de clasificar.

En consecuencia, los esfuerzos por automatizar el reconocimiento de especies de aves mediante imágenes a nivel nacional han sido limitados. En su mayoría, los estudios se han centrado en monitoreo manual o en la identificación tradicional de especies mediante observación directa, lo cual implica una gran carga de trabajo y depende de la experiencia y conocimientos del profesional a cargo. Todo este material recopilado no se almacena adecuadamente, generando repositorios locales con alta variabilidad, baja calidad y sin las etiquetas correspondientes. Por esta razón, se dificultan aún más el desarrollo de modelos de clasificación precisos para aves locales y se tiende a usar herramientas externas para este tipo de tareas.

1.6 Alcance

El presente trabajo se enfoca en el desarrollo de un sistema de aprendizaje automático para la clasificación de aves de la Región Andina del Ecuador, utilizando técnicas de visión por computadora. El proyecto se guiará bajo las 6 fases del marco de trabajo CRISP-DM. Cabe aclarar que se considerarán únicamente aves cuya presencia esté documentada en la Región Andina ecuatoriana, trabajando con un conjunto limitado de especies previamente seleccionadas.

El modelo se evaluará desde distintos enfoques e incluirá la integración de una capa de visualización que permita interpretar los resultados de la clasificación de manera sencilla, no se contempla en esta etapa la implementación de una aplicación móvil o web funcional para el público general, aunque se dejarán sentadas las bases técnicas para desarrollos futuros en ese sentido.

CAPÍTULO II – MARCO TEÓRICO

2.1. Inteligencia Artificial

La Inteligencia Artificial es un área de la informática dedicada a diseñar sistemas capaces de emular funciones cognitivas humanas como el aprendizaje, el razonamiento y la percepción. Su propósito es crear máquinas que puedan resolver problemas complejos y adaptarse a entornos cambiantes sin intervención humana directa. El trabajo desde sus primeras manifestaciones ha evolucionado hacia un campo interdisciplinario que integra conocimientos de estadística, neurociencia, lógica y ciencias computacionales, permitiendo que algoritmos inteligentes transformen procesos en sectores como la industria, la salud y la educación (Russell & Norvig, 2010).

A lo largo de su desarrollo, la IA se ha estructurado en distintos enfoques. Una de las más significativas es la IA conexionista, basada en modelos inspirados en la estructura del cerebro humano, como las redes neuronales artificiales. Su especial relevancia va de la mano con el auge del aprendizaje automático y el aprendizaje profundo, al demostrar un notable desempeño en tareas de clasificación de imágenes, procesamiento del lenguaje natural y predicciones complejas (Goodfellow et al., 2016).

En la actualidad, la Inteligencia Artificial se considera un pilar fundamental en la creación de sistemas autónomos que optimizan procesos y generan conocimiento a partir de grandes volúmenes de datos. La capacidad de los algoritmos para aprender patrones y mejorar su rendimiento de manera progresiva ha impulsado la investigación y la aplicación de soluciones innovadoras en múltiples disciplinas científicas y tecnológicas (Russell & Norvig, 2010).

2.2 Aprendizaje Automático y Deep Learning

El Aprendizaje Automático es una rama esencial de la Inteligencia Artificial que dota a los sistemas de la capacidad de identificar patrones y mejorar su rendimiento a partir de datos, sin ser programados de manera explícita. Sus métodos permiten construir modelos predictivos capaces de adaptarse dinámicamente conforme procesan nuevas observaciones. Dentro de este ámbito, el Deep Learning destaca por emplear redes neuronales con múltiples capas que modelan transformaciones jerárquicas de los datos, logrando un rendimiento superior en tareas complejas como el reconocimiento de imágenes y el procesamiento del lenguaje natural (LeCun, Bengio, & Hinton, 2015).

En este contexto, es oportuno decir que la IA abarca el desarrollo de programas con la habilidad de razonar como humanos, mientras que el Machine Learning se centra en algoritmos que aprenden de la experiencia sin programación explícita. Asimismo, dentro de este subconjunto, el Deep Learning se diferencia por su capacidad de procesar grandes volúmenes de datos mediante redes neuronales que se adaptan automáticamente. Gracias a estas capacidades, hoy es posible abordar problemas que requieren procesar grandes volúmenes de datos y descubrir patrones con un nivel de precisión sin precedentes.

Figura 1

Inteligencia Artificial, Machine Learning y Deep Learning

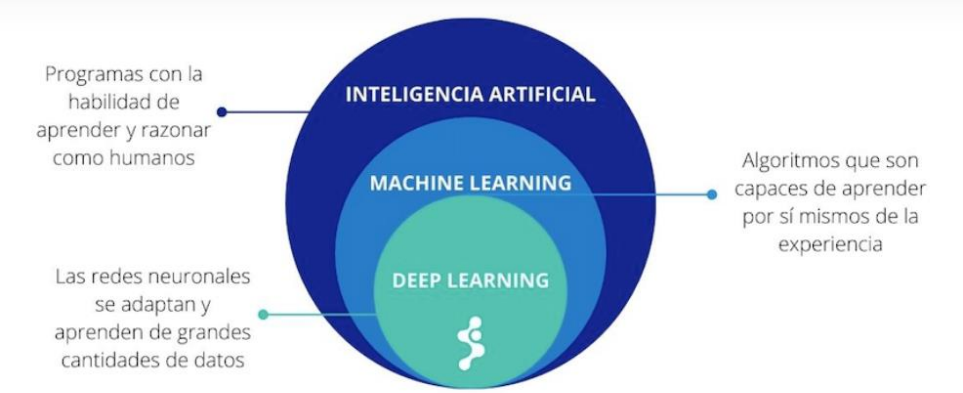


Figura 1: Inteligencia Artificial, Machine Learning y Deep Learning (IASolver, s.f.)

2.3 Redes Neuronales

Las redes neuronales son modelos computacionales inspirados en el funcionamiento de las neuronas biológicas, diseñados para procesar información mediante la interconexión de unidades simples llamadas neuronas artificiales. Estas redes están estructuradas en capas, donde una capa de entrada recibe los datos, una o varias capas ocultas transforman las señales mediante funciones de activación no lineales, y una capa de salida que produce la predicción final. Gracias a este proceso, las redes neuronales pueden aprender representaciones complejas de los datos a partir de ejemplos y ajustar sus parámetros internos mediante algoritmos de optimización (Goodfellow, Bengio, & Courville, 2016).

El auge de las redes neuronales se ha consolidado con el desarrollo del aprendizaje profundo, donde arquitecturas con numerosas capas permiten modelar relaciones de alta complejidad en problemas de visión por computadora, procesamiento del lenguaje y análisis de

series temporales. Estas redes profundas, conocidas como Deep Neural Networks, requieren grandes volúmenes de datos y capacidad de cómputo considerable, pero han demostrado superar los métodos tradicionales en precisión y capacidad de generalización. Su éxito radica en la habilidad para extraer características jerárquicas de la información y adaptar su estructura a la naturaleza de los datos, constituyendo así el núcleo de muchas aplicaciones modernas de inteligencia artificial (Schmidhuber, 2015).

2.3.1 Redes Neuronales Convolucionales

Las CNN constituyen un tipo especializado de arquitectura diseñada específicamente para procesar datos con una estructura espacial, como imágenes o secuencias bidimensionales. A diferencia de las redes neuronales tradicionales que conectan cada neurona con todas las entradas, las convolucionadas utilizan filtros que se desplazan sobre la imagen para detectar patrones locales, como bordes, texturas y formas. Esta propiedad de aprendizaje jerárquico permite extraer características relevantes con un nivel creciente de abstracción, lo que las convierte en una herramienta fundamental para tareas de reconocimiento visual (Goodfellow, Bengio, & Courville, 2016).

La estructura básica de una CNN está compuesta por capas convolucionales que aplican los filtros y generan mapas de activación, seguidas de capas de pooling que reducen la dimensionalidad conservando la información más importante. Posteriormente, los datos son aplanados y procesados por capas completamente conectadas que combinan las características aprendidas y producen las predicciones finales. Este flujo permite que el modelo identifique patrones complejos en imágenes de alta resolución y generalice eficazmente a nuevos ejemplos no vistos durante el entrenamiento (LeCun, Bengio, & Hinton, 2015).

Figura 2

Funcionamiento de una Red Neuronal Convolucional

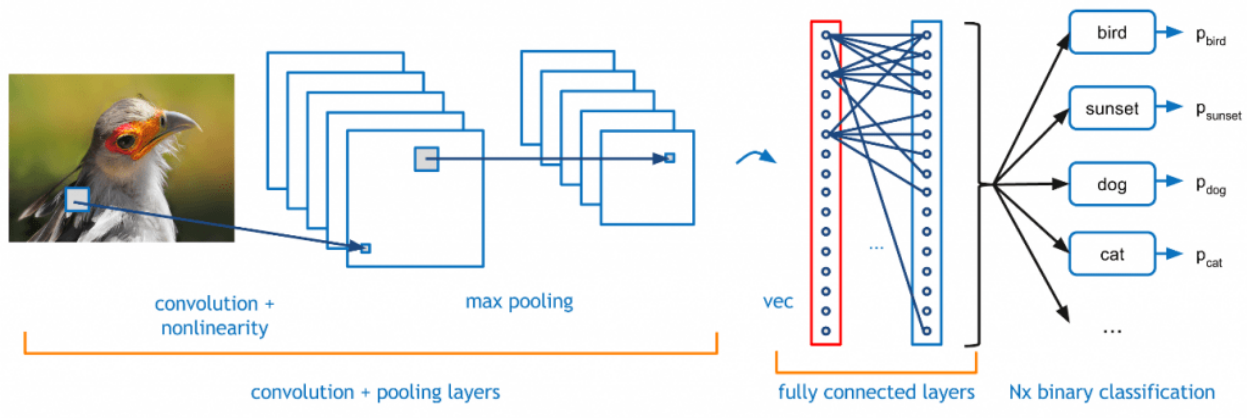


Figura 2: Funcionamiento de una Red Neuronal Convolucional (LIS Data Solutions, s.f.)

2.4 Representación Cromática

Para trabajo con imágenes, la representación cromática es un aspecto fundamental del procesamiento, ya que define la manera en que los colores son capturados, almacenados y procesados por los sistemas de visión artificial. Las imágenes pueden representarse mediante diferentes espacios de color, siendo el más utilizado el modelo RGB (Red, Green, Blue), donde cada píxel se describe mediante la combinación de intensidades de los tres canales primarios de luz. Esta representación permite conservar la información visual completa de la escena y facilita que los modelos de aprendizaje profundo identifiquen patrones cromáticos distintivos que contribuyen a la clasificación de objetos (González & Woods, 2018).

En el contexto de clasificación de aves, la información cromática adquiere especial relevancia porque el color del plumaje constituye una característica discriminativa clave. No obstante, en ciertos casos se utilizan representaciones alternativas como la escala de grises, que

prescinde de la información de color y conserva únicamente la intensidad luminosa de cada píxel. Esta simplificación puede ser útil para reducir la complejidad computacional, aunque generalmente disminuye la precisión del modelo al eliminar matices importantes de la imagen (O'Gorman & Kasturi, 2019).

2.5 Metodología CRISP-DM

El marco de trabajo CRISP-DM (Cross Industry Standard Process for Data Mining) es un modelo de referencia ampliamente utilizado para estructurar proyectos de minería de datos y aprendizaje automático. Propone seis fases iterativas: la comprensión del negocio, que consiste en definir los objetivos y requisitos desde una perspectiva organizacional; la comprensión de los datos, orientada a recolectar, explorar y familiarizarse con la información disponible; la preparación de los datos, que abarca las tareas de limpieza, transformación y selección de atributos relevantes; el modelado, donde se eligen y aplican algoritmos de aprendizaje; la evaluación, que revisa si los modelos cumplen con los objetivos planteados; y finalmente el despliegue, que implementa la solución en un entorno real de aplicación (IBM, 2020).

La metodología posee la ventaja de que cada fase puede retroalimentar a las anteriores para realizar ajustes conforme se obtienen nuevos hallazgos. Esta flexibilidad permite adaptar el proceso a distintos dominios y tipos de proyectos, garantizando que los resultados sean reproducibles, alineados con los objetivos estratégicos y útiles para la toma de decisiones. Su aplicación resulta valiosa en proyectos de Deep Learning, donde la experimentación con diferentes arquitecturas de redes neuronales, técnicas de preprocesamiento y estrategias de evaluación requiere un enfoque iterativo que permita optimizar progresivamente la calidad del modelo y su capacidad de generalización.

Figura 3

Ciclo de vida metodología CRISP-DM

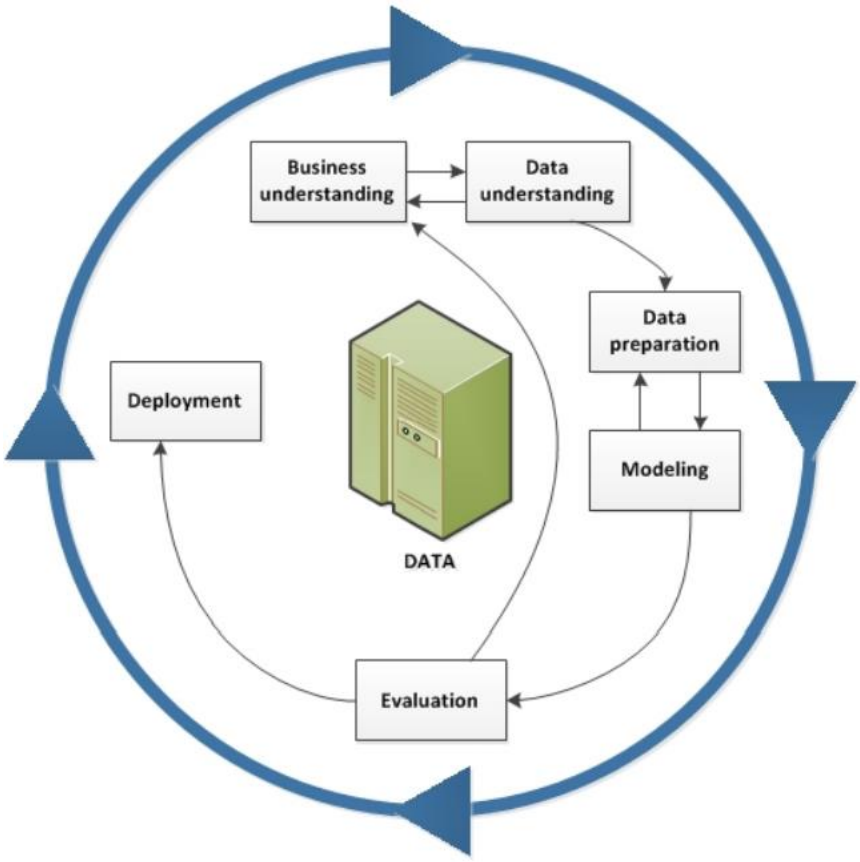


Figura 3: Ciclo de vida metodología CRISP-DM (IBM, 2020)

2.6. Métricas de Evaluación de Modelos de Clasificación

La evaluación de los modelos de clasificación es una etapa fundamental para determinar su rendimiento y su capacidad de generalizar a nuevos datos. Entre las métricas más utilizadas se encuentra el "accuracy", que indica la proporción de predicciones correctas sobre el total de casos. Sin embargo, en problemas con clases desbalanceadas, este indicador puede ser engañoso, por lo que resulta habitual complementarlo con el "precisión", que mide la proporción de verdaderos

positivos respecto al total de predicciones positivas realizadas, y el "recall", que refleja la capacidad del modelo para identificar todas las instancias positivas reales (Fawcett, 2006).

Otra métrica complementaria es el "F1-Score", que corresponde a la media armónica entre la precisión y el recall, ofreciendo un balance adecuado cuando existe un compromiso entre ambos. Además, la matriz de confusión permite visualizar el comportamiento detallado del clasificador al mostrar la distribución de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos, facilitando el análisis de los errores más frecuentes (Sokolova & Lapalme, 2009).

2.7 Tecnologías y Herramientas

El desarrollo de un sistema de clasificación basado en aprendizaje profundo requiere integrar diversas tecnologías que faciliten tanto la creación y entrenamiento de modelos como su posterior despliegue en un prototipo funcional. Existen ya en el mercado herramientas estándar que están enfocadas en el desarrollo de proyectos de Deep Learning, como TensorFlow y Keras para la definición y optimización de redes neuronales convolucionales, Django como framework web orientado a la creación de interfaces de usuario interactivas, y librerías de soporte como NumPy y Matplotlib para el procesamiento numérico y la visualización de resultados. La combinación de estas tecnologías permite implementar soluciones robustas y escalables que garanticen que los modelos desarrollados puedan integrarse de forma eficaz en aplicaciones prácticas orientadas a la clasificación automática de imágenes.

2.7.1. TensorFlow y Keras

TensorFlow es una biblioteca de código abierto desarrollada por Google que facilita la construcción y entrenamiento de modelos de aprendizaje automático a gran escala. Su arquitectura

permite definir grafos de operaciones matemáticas que se ejecutan de manera eficiente en CPUs y GPUs, ofreciendo un alto rendimiento en cálculos matriciales intensivos, característicos de los algoritmos de aprendizaje profundo. Su diseño modular permite que se adapte a proyectos de investigación, prototipado y entornos de producción, permitiendo implementar modelos a cualquier grado de complejidad (Abadi et al., 2016).

Sobre TensorFlow se integra Keras, una API de alto nivel que simplifica la creación de redes neuronales mediante un enfoque declarativo y flexible. Se encarga de proporcionar componentes predefinidos como capas convolucionales, funciones de activación, optimizadores y herramientas para la preparación de datos, lo que reduce significativamente el tiempo de desarrollo. Esta ventaja la ha convertido en una de las interfaces más populares para investigadores y profesionales que buscan construir modelos robustos con una curva de aprendizaje accesible (Chollet, 2018).

Las herramientas en conjunto ofrecen un ecosistema completo que favorece la experimentación, la monitorización del entrenamiento y la exportación de modelos a diferentes plataformas. Esta combinación ha sido determinante en proyectos de clasificación de imágenes, ya que permite gestionar grandes volúmenes de datos, realizar ajustes de hiperparámetros y desplegar modelos entrenados en aplicaciones web o móviles con eficiencia y escalabilidad (Abadi et al., 2016).

2.7.2. Figma

Se define como una plataforma de diseño colaborativo basada en la nube que permite crear interfaces de usuario, prototipos interactivos y sistemas de diseño de forma centralizada. Su arquitectura facilita que equipos multidisciplinarios trabajen simultáneamente en un mismo proyecto, lo que agiliza los ciclos de diseño y revisión. Entre sus funcionalidades se destacan la creación de componentes reutilizables, la integración con bibliotecas de estilos y la posibilidad de compartir prototipos en tiempo real con desarrolladores, fomentando la transparencia y la coherencia visual en todas las etapas del desarrollo (Figma, s.f.).

Para cualquier tipo de proyecto que requiera la elaboración de una interfaz de usuario, resulta útil la herramienta para diseñar interfaces intuitivas que faciliten la interacción y presenten un grado de facilidad alto para corregir componentes antes de implementarlos en código. Su capacidad de generar especificaciones técnicas y recursos gráficos exportables permite optimizar la integración entre el diseño de la experiencia de usuario y la capa funcional de las aplicaciones.

2.7.3. Django

Django es un framework de desarrollo web de alto nivel, escrito en Python, que facilita la creación de aplicaciones robustas mediante un enfoque basado en la reutilización de componentes y la automatización de tareas comunes. Entre sus principales características se encuentran un sistema de enrutamiento flexible, un ORM para la gestión de bases de datos y un conjunto de utilidades que simplifican la implementación de autenticación, seguridad y administración de contenidos.

Uno de los aspectos más relevantes de framework es su patrón arquitectónico MVT, en donde el Modelo se encarga de definir la estructura de la base de datos y las reglas de negocio, la Vista contiene la lógica que responde a las solicitudes del usuario y prepara los datos, y el Template se ocupa de renderizar el contenido en formato HTML. Es especialmente adecuado para proyectos que integran modelos de aprendizaje automático y aprendizaje profundo, ya que proporciona mecanismos seguros y escalables para exponer las predicciones mediante interfaces web. Gracias a su compatibilidad con bibliotecas de Python y su arquitectura modular, es posible encapsular modelos entrenados en vistas que reciben datos, procesan la información y devuelven resultados en tiempo real (Django Software Foundation, s.f.).

Figura 4

Funcionamiento MTV de Django

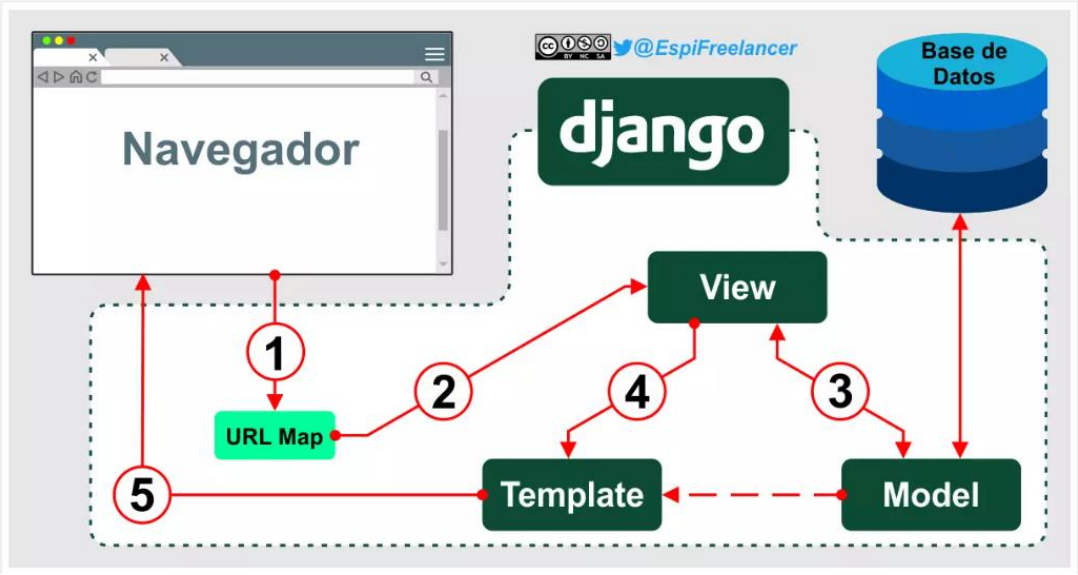


Figura 4: Funcionamiento MTV de Django (EspiFreelancer, s.f.)

CAPÍTULO III – METODOLOGÍA

3.1 Investigación Científica

El presente trabajo se fundamenta por ser de carácter científico aplicada, orientado a la generación de conocimiento mediante el desarrollo de una solución práctica en el campo del Deep Learning, respondiendo a la necesidad de herramientas tecnológicas para el reconocimiento automático de aves de la zona andina del Ecuador a partir de imágenes fotográficas. Este proceso involucra la recolección, análisis y normalización de datos válidos, así como la construcción, entrenamiento, prueba y evaluación de modelos basados en redes neuronales para asegurar el cumplimiento de los criterios mínimos de validez. De esta manera, se garantiza la objetividad y confiabilidad de los resultados, elementos esenciales para aportar soluciones tecnológicas que contribuyan al desarrollo académico y a su integración en nuevos sistemas en el ámbito ambiental.

3.1.1 Enfoque y Tipo de Investigación

El enfoque de implementación es de tipo cuantitativo, el cual permite evaluar objetivamente la precisión y eficacia del modelo a desarrollar a través de métricas estadísticas y experimentos controlados. En cuanto al tipo de investigación, se clasificó principalmente como experimental, dado que se diseñaron y ejecutaron pruebas controladas sobre conjuntos de datos previamente preparados para validar y comparar distintas configuraciones y enfoques de las redes neuronales. Además, es de naturaleza descriptiva y exploratoria, al analizar y reportar las características y resultados de cada modelo, en la búsqueda de nuevas técnicas para optimizar la clasificación automática.

3.1.2 Población y Muestra

La muestra de imágenes para el desarrollo del sistema está constituida principalmente de dos fuentes confiables: repositorio privado del World Conservation Society (WCS) Ecuador y repositorios de la plataforma social iNaturalist. Estas bases proporcionan una gran cantidad de imágenes para fines académicos y metadatos validados, que permiten garantizar la calidad, diversidad y representatividad visual necesarias para el entrenamiento y evaluación del modelo. "La región andina ecuatoriana alberga cerca de 600 especies de aves, constituyendo una de las áreas con mayor diversidad ornitológica en el país, lo que resalta su relevancia para la conservación y el estudio científico" (Freile & Cuervo, 2019).

Dada la magnitud y gran variedad de especies, se debe limitar el desarrollo a utilizar cinco especies representativas de la zona, las cuales son:

- **Clarinero Escarlata (Anisognathus igniventris):** Ave perteneciente a la familia Thraupidae, reconocida por su plumaje rojo vibrante en el abdomen. Habita principalmente en bosques montanos húmedos de las provincias de Pichincha, Cotopaxi y Tungurahua (Ridgely & Greenfield, 2001).

Figura 5

Clarinero Escarlata (Anisognathus igniventris)



Figura 5: Clarinero Escarlata (Anisognathus igniventris) (© Nicole LaRoche)

- **Carpintero Candela Común (Colaptes rivolii):** Carpintero mediano caracterizado por un patrón barrado en su plumaje, frecuente en bosques montanos y páramos andinos de Imbabura y Carchi (Ridgely & Greenfield, 2001).

Figura 6

Carpintero Candela Común (Colaptes rivolii)



Figura 6: Carpintero Candela Común (Colaptes rivolii) (© Karl Kroeker)

- **Águila Pechinegra (Geranoaetus melanoleucus):** Ave rapaz robusta que se distribuye en áreas abiertas y montañosas de las provincias de Azuay, Loja y Chimborazo (Ridgely & Greenfield, 2001).

Figura 7

Águila Pechinegra (Geranoaetus melanoleucus)



Figura 7: Águila Pechinegra (Geranoaetus melanoleucus) (© Sebastián Lescano)

- **Pava Andina (Penelope montagnii):** Ave que habita principalmente en bosques montanos húmedos de la región andina, especialmente en las provincias de Tungurahua, Pichincha, Carchi y Azuay. Se caracteriza por su plumaje marrón moteado y su tamaño considerable, siendo una especie emblemática de estas zonas elevadas (Ridgely & Greenfield, 2001).

Figura 8

Pava Andina (Penelope montagnii)



Figura 8: Pava Andina (Penelope montagnii) (© cmmteixeira)

- **Cóndor Andino (Vultur gryphus):** Ave carroñera emblemática con amplia envergadura, que se encuentra principalmente en páramos y zonas altas de Cotopaxi, Chimborazo y Pichincha (Ridgely & Greenfield, 2001).

Figura 9

Cóndor Andino (Vultur gryphus)



Figura 9: Cóndor Andino (Vultur gryphus) (© Luis Enrique Calderón Franco)

El enfoque científico aplicado aseguró un proceso estructurado y riguroso que permitió la generación de resultados válidos y confiables. La combinación de metodologías cuantitativas, diseño experimental y análisis estadístico permitió un desarrollo efectivo del sistema de clasificación, conjuntamente con la contribución significativa al conocimiento en el área de la conservación ambiental mediante tecnologías de aprendizaje automático.

3.2 Investigación Técnica

La investigación técnica en este proyecto se centró en los procesos para el procesamiento de datos, desarrollo, entrenamiento y evaluación de un sistema automatizado para la clasificación de aves de la región andina del Ecuador mediante técnicas avanzadas de aprendizaje profundo. Este proceso implicó la selección y preparación rigurosa de imágenes útiles, la construcción de modelos de redes neuronales convolucionales, enfoques del modelo y la implementación de técnicas específicas para mejorar su precisión y robustez. Asimismo, incluye la validación cuidadosa de los resultados mediante métricas cuantitativas que permitan comparar y optimizar las distintas configuraciones aplicadas al rendimiento. Para organizar y ejecutar eficientemente estas actividades, se adoptó un marco metodológico estructurado e iterativo que guió el trabajo en fases definidas, asegurando un desarrollo coherente y sistemático del proyecto.

3.2.1 Enfoque Metodológico CRISP-DM

Para la correcta ejecución técnica del sistema de clasificación de aves se adoptó el marco CRISP-DM, que organiza el proceso en seis fases interrelacionadas. Este enfoque facilita la planificación, ejecución y control del proyecto, permitiendo ajustar y mejorar el modelo en cada

etapa según los resultados obtenidos. A continuación, se describen las fases y su aplicación específica en el presente trabajo.

- **Comprensión del Negocio:** Fase inicial para recolectar los requerimientos funcionales y técnicos necesarios para el desarrollo del sistema de clasificación automática de aves. Establecer KPIs para evaluar la efectividad del modelo y su éxito. Además, analizar limitaciones existentes relacionadas con la calidad, cantidad y disponibilidad de los datos, considerando cómo estos factores podrían influir en las etapas posteriores de entrenamiento y validación del modelo.

- **Comprensión de los Datos:** Fase destinada a un análisis exploratorio inicial de los conjuntos de imágenes disponibles para el proyecto, con el fin de evaluar su calidad, diversidad y representatividad en relación con las especies seleccionadas. Se identifican posibles problemas como datos incompletos, imágenes de baja resolución o etiquetado incorrecto que requieran tratamiento. Además, se realiza una evaluación preliminar de las características relevantes y estadística descriptiva de los datos, que puedan influir en las fases posteriores.

- **Preparación de los Datos:** En esta fase se contempla el procesamiento y transformación de las imágenes seleccionadas para adecuarlas a los requerimientos del modelo convolucional. Incluye aspectos como el filtrado, redimensionamiento, normalización, etiquetado y la generación de nuevas muestras mediante técnicas de aumento de datos para

mejorar la diversidad del conjunto. Además, se aplican criterios que permitan garantizar la generalización y robustez del modelo durante su desarrollo.

- **Modelado:** Etapa de construcción y entrenamiento de los modelos de redes neuronales convolucionales para la clasificación de imágenes. Se seleccionan las arquitecturas y parámetros iniciales del modelo, incluyendo la configuración de capas, funciones de activación y técnicas de regularización como DropOut. Además, el desarrollo de tres modelos distintos para comparar su desempeño: utilizando imágenes en color RGB, escala de grises y realizar la optimización del mejor modelo obtenido mediante ajuste de hiperparámetros y técnicas de regularización avanzadas. El entrenamiento se realiza con los conjuntos de datos preparados, y se monitorizará cuidadosamente el comportamiento de cada modelo para identificar la configuración más eficiente.

- **Evaluación:** La evaluación del desempeño de los modelos se realiza utilizando métricas cuantitativas globales y específicas para cada clase o especie, incluyendo precisión (precision), sensibilidad (recall), y puntaje F1 (f1-score). Además, se analiza la matriz de confusión para identificar patrones de clasificación errónea y entender mejor el comportamiento del modelo frente a cada categoría. Estas métricas permiten evaluar el rendimiento general y el desempeño particular en la clasificación de cada especie, lo que es crucial dada la variabilidad visual entre las aves estudiadas. Para facilitar el seguimiento y comparación de los modelos, se utiliza TensorBoard como herramienta de visualización de las métricas y análisis de comportamiento de los modelos propuestos.

- **Despliegue:** Se implementa una capa de visualización que permita cargar el modelo con mejor desempeño y realizar pruebas de clasificación de imágenes de aves con nuevas fotos. Esta interfaz no está montada en un servidor y funciona como una herramienta local destinada a mostrar los resultados del modelo de forma clara y comprensible. Siendo útil como recurso educativo y de apoyo para actividades ambientales que requieran la identificación y clasificación de especies. Esta capa puede ser integrada posteriormente en plataformas más complejas o aplicaciones móviles, ampliando su alcance y funcionalidad.

CAPÍTULO IV – DESARROLLO DEL SISTEMA DE APRENDIZAJE AUTOMÁTICO

4.1 Organización y Preparación del Conjunto de Datos de Aves

Para el desarrollo exitoso del proyecto, fue fundamental disponer de un conjunto de datos organizado y etiquetado de manera precisa que permitiera un entrenamiento efectivo del modelo y garantice resultados óptimos. En las fases iniciales del marco CRISP-DM se priorizó la contextualización del área de estudio, así como la comprensión profunda de los datos extraídos y de las características esenciales que influyen en el desempeño del sistema. Dado que CRISP-DM es un proceso iterativo, se contempló la posibilidad de repetir y ajustar las etapas de análisis y preparación de datos hasta identificar la mejor estrategia para el manejo y transformación del conjunto de imágenes, optimizando así la calidad y relevancia de la información utilizada para el entrenamiento del modelo.

4.1.1 Entendimiento del Negocio

Al tratarse de un proyecto sin un enfoque comercial directo, no fue posible definir un objetivo comercial acorde a la naturaleza de la organización que vaya alineada a una evaluación de la situación actual. Estos aspectos no se abordaron directamente pero el producto busca la futura implementación en el ámbito científico-académico de toda empresa interesada en la biodiversidad, proyectos de conservación y estudio de fauna local.

En este contexto, existen repositorios y galerías de biólogos que no han sido aprovechados para construir una herramienta que beneficie a la comunidad interesada estudio de las aves. Por lo tanto, el objetivo de minería se definió de la siguiente manera:

- Clasificar automáticamente especies representativas de aves de la Región Andina del Ecuador a partir de imágenes recolectadas en repositorios digitales.

Para evaluar éxito del sistema, se establecen los siguientes indicadores clave de desempeño:

- **Umbral de precisión de 75%:** Garantizar que el modelo alcance resultados confiables y aplicables.
- **Tiempo de procesamiento por imagen:** Medir la rapidez con la que el sistema clasifica cada imagen.
- **Capacidad de generalización:** Evaluar la habilidad del modelo para mantener un buen rendimiento en imágenes nuevas y variadas.

4.1.2 Entendimiento de los Datos

Existen múltiples repositorios y galerías que ofrecen fotografías de alta calidad para el desarrollo del proyecto. En cuestión, se decidió optar por dos fuentes de datos con contenido diverso y representativo de las especies de aves de la Región Andina del Ecuador. Esta selección garantizó la construcción de un conjunto de imágenes sólida, lo que resulta indispensable en la obtención de resultados favorables en las fases posteriores.

Como principal fuente de datos se utilizó la red social iNaturalist, reconocida por su extenso catálogo de imágenes de alta calidad destinadas a usos académicos y científicos. La extracción de imágenes se completó mediante su API, la cual permitió definir parámetros específicos de búsqueda para obtener fotografías adecuadas y relevantes para la construcción del modelo. Complementariamente, se empleó la galería privada de WCS Ecuador, cuyos biólogos e

investigadores han aportado un valioso conjunto de imágenes seleccionadas y autorizadas para su uso exclusivo en este proyecto.

4.1.2.1 Extracción de Datos API iNaturalist. iNaturalist es una plataforma social de ciencia y divulgación que ofrece una extensa base de datos de observaciones naturales, incluyendo millones de fotografías con anotaciones y metadatos validados por expertos. Esta plataforma es ampliamente utilizada para fines académicos y científicos debido a la calidad y diversidad de su contenido, siendo un recurso fundamental para proyectos de biodiversidad y monitoreo ambiental.

Para la obtención masiva y automática de imágenes se hizo uso del script de Python "inat_downloader" el cual permite conectarse a la API de iNaturalist y descargar imágenes y metadatos desde el repositorio de observaciones. La herramienta permite configurar parámetros para adaptar a las necesidades de todo tipo de proyecto o investigación.

El repositorio oficial es de acceso público y de código abierto, distribuido bajo la licencia MIT, permitiendo su uso, modificación y distribución siempre que se respeten los límites impuestos por la API de iNaturalist. Acceso directo al repositorio, véase el Anexo 1.

Para configurar la descarga fue necesario clonar el repositorio y preparar un entorno de trabajo, seguido de la instalación de la biblioteca "requests", la cual permite enviar solicitudes HTTP/HTTPS a la API de iNaturalist y extraer los datos. Posteriormente, fue fundamental contar con un archivo CSV con el nombre de "species.csv", el cual contiene los nombres científicos de las cinco especies seleccionadas para el proyecto y un identificador inicial para la descarga. Esta estructura es necesaria para controlar las observaciones de descarga y poder reanudar el proceso en caso de interrupciones.

Tabla 1

Especies seleccionadas para la descarga de observaciones desde iNaturalist.

name	start_id
Anisognathus igniventris	0
Colaptes rivolii	0
Geranoaetus melanoleucus	0
Penelope montagnii	0
Vultur gryphus	0

Tabla 1: Especies seleccionadas para la descarga de observaciones desde iNaturalist

Para la ejecución del script fue necesario configurar ciertos parámetros clave para obtener resultados válidos, para este proyecto se usaron los siguientes:

- **Número de imágenes por especie:** 600
- **Calidad de observación:** research
- **Licencias permitidas:** cc-by, cc-by-nc, cc0
- **Tamaño de las imágenes:** original

Figura 10

Ejecución del script para la descarga masiva de imágenes desde iNaturalist



Figura 10: Ejecución del script para la descarga masiva de imágenes desde iNaturalist

El funcionamiento del script cuenta con tres partes fundamentales. La función evaluate_query_rate() monitorea cuántas consultas se han realizado a la API de iNaturalist en el día. Si se supera el límite permitido, el script pausa la ejecución y espera hasta que el contador se reinicie, evitando así que el servicio bloquee el acceso por exceso de solicitudes. La función download() fue la encargada de procesar cada observación obtenida desde la API, extrayendo metadatos relevantes como nombre de especie, ID de observación, licencia, usuario, calidad, fecha y ubicación. Estos datos fueron almacenados en un archivo CSV específico para cada especie, facilitando su organización y análisis posterior. Paralelamente, la función descargó todas las imágenes asociadas a cada observación, ajustando su tamaño según lo configurado, y las guarda en carpetas organizadas por especie con nombres que incluyen información clave para su identificación. Por último, el script leyó el archivo CSV con la lista de especies a descargar y automatiza el proceso para cada una, creando las carpetas necesarias y gestionando la descarga de imágenes y metadatos respetando los límites y la estructura establecida.

Es importante destacar que el parámetro que indica la cantidad de observaciones a descargar puede no coincidir exactamente con el número final de imágenes obtenidas. La razón es que cada observación puede contener múltiples fotografías, y el script contabiliza una observación por entrada en el conjunto de datos, no por la cantidad total de imágenes. Por lo tanto, la cantidad total de imágenes descargadas puede superar el número de observaciones solicitado. Al concluir el proceso, se obtuvo un conjunto organizado de archivos que contienen las imágenes, junto con un CSV independiente por especie que almacenan los metadatos, una vez los datos recolectados estuvieron listos se pudo continuar con los pasos iniciales de exploración.

Figura 11

Función para la extracción de imágenes y metadatos desde la API iNaturalist

```
def download(my_species_name, observations, image_size):
    global current_observations_number, current_images_number, current_dataset_size

    for observation in observations:
        species_name = observation["taxon"]["name"]
        observation_id = observation["id"]
        observation_license = observation.get("license_code", "none")
        observer_login = observation["user"]["login"]
        observation_quality = observation.get("quality_grade", "none")
        observation_date = observation.get("observed_on", "none")
        if observation.get("geojson"):
            observation_latitude = observation["geojson"]["coordinates"][1] or "none"
            observation_longitude = observation["geojson"]["coordinates"][0] or "none"
        else:
            observation_latitude = "none"
            observation_longitude = "none"

        current_observations_number += 1
        print(f"INFO : {my_species_name} - Observation
{current_observations_number}/{max_observations_number} (ID : {observation_id})")

        with open(f"results/{my_species_name.replace(' ', '_')}_metadata.csv", 'a', newline='') as
csvfile:
            csv_writer = csv.writer(csvfile)
            csv_writer.writerow([
                species_name,
                observation_id,
                observation_license,
                observer_login,
                observation_quality,
                observation_date,
                observation_latitude,
                observation_longitude
            ])

        for photo_id, photo in enumerate(observation["photos"]):
            image_url = photo["url"].replace("/square", f"/{image_size}")
            file_name = f"{species_name.replace(' ','-
')}_{observer_login}_{observation_license}_{observation_id}_{photo_id}.jpeg"
            file_path = os.path.join(f"results/{my_species_name.replace(' ', '_')}_images", file_name)

            image_response = session.get(image_url)
            if image_response.status_code == 200:
                with open(file_path, "wb") as image_file:
                    image_file.write(image_response.content)
                    current_images_number += 1
                    current_dataset_size += len(image_response.content) / 1_000_000
                    print(f"INFO : {current_images_number} images downloaded
({round(current_dataset_size, 2)} MB)")

                    my_hourly_media["value"] += len(image_response.content) / 1_000_000_000
                    my_daily_media["value"] += len(image_response.content) / 1_000_000_000
                    evaluate_media_rate()
            else:
                print(f"WARNING : Couldn't download image at {image_url}")

    return observation_id
```

Figura 11: Función para la extracción de imágenes y metadatos desde la API iNaturalist

4.1.2.2 Análisis Exploratorio y Estadístico de Imágenes. Como parte del proceso de exploración inicial, fue necesario comprender la estructura, distribución y características del conjunto de datos previamente extraído. Este análisis permitió identificar la calidad y diversidad de las fotografías, así como los problemas y desequilibrios que requieran ajustes en fases posteriores. Para mejorar la comprensión se usó gráficos de visualización que permiten interpretar las características del conjunto de datos.

El primer paso consistió en cuantificar el número de imágenes obtenidas para cada una de las cinco especies seleccionadas. Después del conteo se obtuvo un total de 4823 imágenes en total, distribuidas de la siguiente forma en 5 carpetas:

- Águila Pechinegra: 767 imágenes
- Carpintero Candela: 1041 imágenes
- Clarinero Escarlata: 781 imágenes
- Cóndor Andino: 1348 imágenes
- Pava Andina: 886 imágenes

El gráfico presentado en la Figura 12, evidenció la variabilidad en la cantidad de imágenes por especie, esta distribución heterogénea generó el primer punto a tener en cuenta para etapas posteriores, el desequilibrio en la cantidad de muestras. El Cóndor Andino fue la especie con mayor número de imágenes y el Águila Pechinegra con la menor cantidad, la diferencia fue de casi 600 fotos, por lo tanto, se consideró las primeras técnicas que permitieron la mitigación de dicho desequilibrio.

Figura 12

Gráfico de barras de la distribución de imágenes por especie

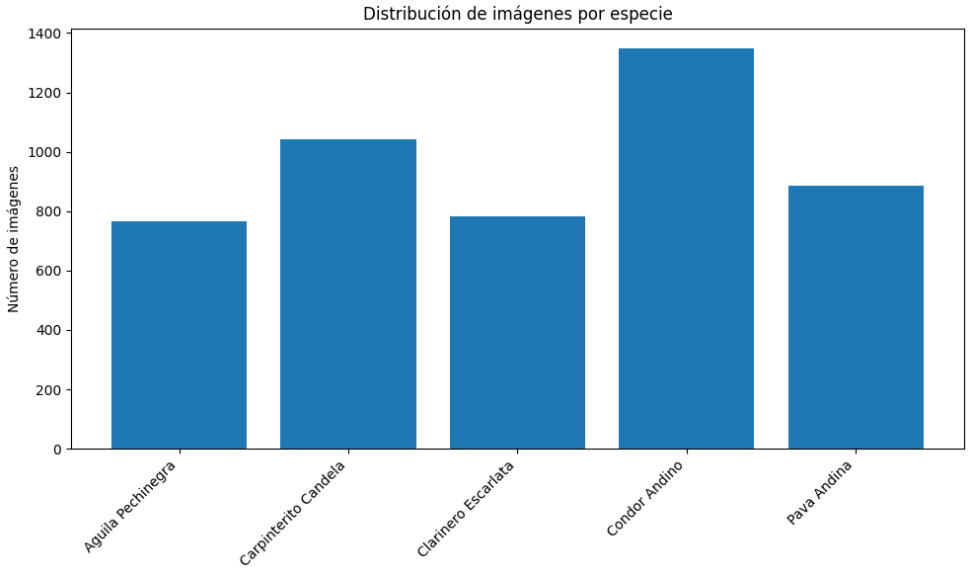


Figura 12: Gráfico de barras de la distribución de imágenes por especie

Continuando el proceso, la heterogeneidad de los datos fue un punto a tomar en cuenta para estandarizar el conjunto de datos y evitar errores en la construcción del modelo, para lo cual, se empleó la librería Pillow (PLI) y se procedió a tabular los resultados con los formatos para cada carpeta.

Tabla 2

Distribución de formatos de imagen por especie

Especie	JPEG	PNG	Formatos Distintos
Águila Pechinegra	754	13	2
Carpintero Candela	1030	11	2
Clarinero Escarlata	776	5	2
Cóndor Andino	1339	9	2
Pava Andina	882	4	2

Tabla 2: Distribución de formatos de imagen por especie

La estadística descriptiva en el conjunto de datos permitió entender el proceso de estandarización requerida en las imágenes. Como primer paso se consideró el análisis de dimensiones en pixeles dentro del conjunto de datos, dado que, los modelos de aprendizaje automático requieren entradas uniformes y redimensionamiento en la mayoría de los casos. Luego del análisis se calculó un promedio del ancho y alto de las imágenes por especie.

Figura 13:

Gráfico de barras de las dimensiones promedio (ancho y alto) por especie

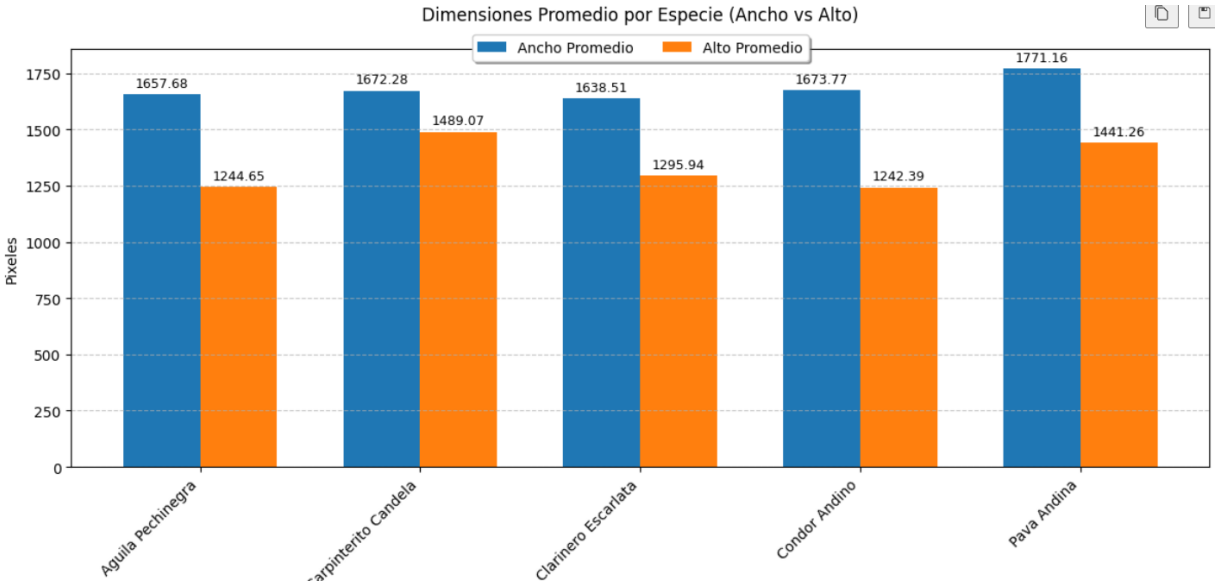


Figura 13: Gráfico de barras de las dimensiones promedio (ancho y alto) por especie

Un aspecto crucial a tomar en cuenta en el análisis de imágenes para modelos de aprendizaje automático es la identificación de los modos de color. Comprender los modos presentes en el conjunto de datos es fundamental para garantizar que todas las imágenes sean procesadas de manera uniforme y no ocasionar problemas con la definición del modelo. Por lo cual, fue necesario identificar los modos de color de cada imagen recolectada y comprender si requieren un proceso de conversión en etapas posteriores.

Tabla 3

Distribución de modos de color de imágenes por especie

Especie	L	RGB	RGBA
Águila Pechinegra	0	762	5
Carpintero Candela	0	1030	11
Clarinero Escarlata	0	778	3
Cóndor Andino	2	1341	5
Pava Andina	0	884	2

Tabla 3: Distribución de modos de color de imágenes por especie

Para complementar la evaluación de la calidad visual de las imágenes se implementó adicionalmente un análisis del brillo y contraste por especie. La variabilidad en estas características puede influir significativamente en la capacidad del modelo para aprender y generalizar características visuales, lo que hace necesario comprender las condiciones de iluminación y tonalidad de las imágenes. El proceso consistió en convertir cada imagen a escala de grises o modo 'L', ya que permite evaluar el brillo como el valor promedio de los píxeles (0-255) y el contraste como la desviación estándar de la distribución de píxeles, sin la complejidad añadida en RGB.

Los resultados de este análisis evidenciaron diferencias notables en el brillo y contraste promedio entre las especies. Las imágenes del Águila Pechinegra y el Cóndor Andino mostraron un brillo promedio más alto (152.51 y 142.36, respectivamente) pero un contraste promedio más bajo (35.14 y 32.57). Esto sugiere que las fotos de estas aves tienden a ser más claras por habitar en ambientes más abiertos y con luz intensa. Por otro lado, especies como la Pava Andina, el Clarinero Escarlata y el Carpinterito Candela presentaron brillos promedio más bajos (entre 114.07 y 120.83) pero contrastes promedio más altos (entre 52.63 y 57.36), dado que las aves habitan generalmente en entornos boscosos o con menos luz directa. Esta heterogeneidad en las

condiciones lumínicas de las imágenes es un factor importante a considerar, ya que podría afectar la capacidad del modelo para extraer características robustas.

Figura 14

Gráfico de barras de brillo y contraste promedio por especie

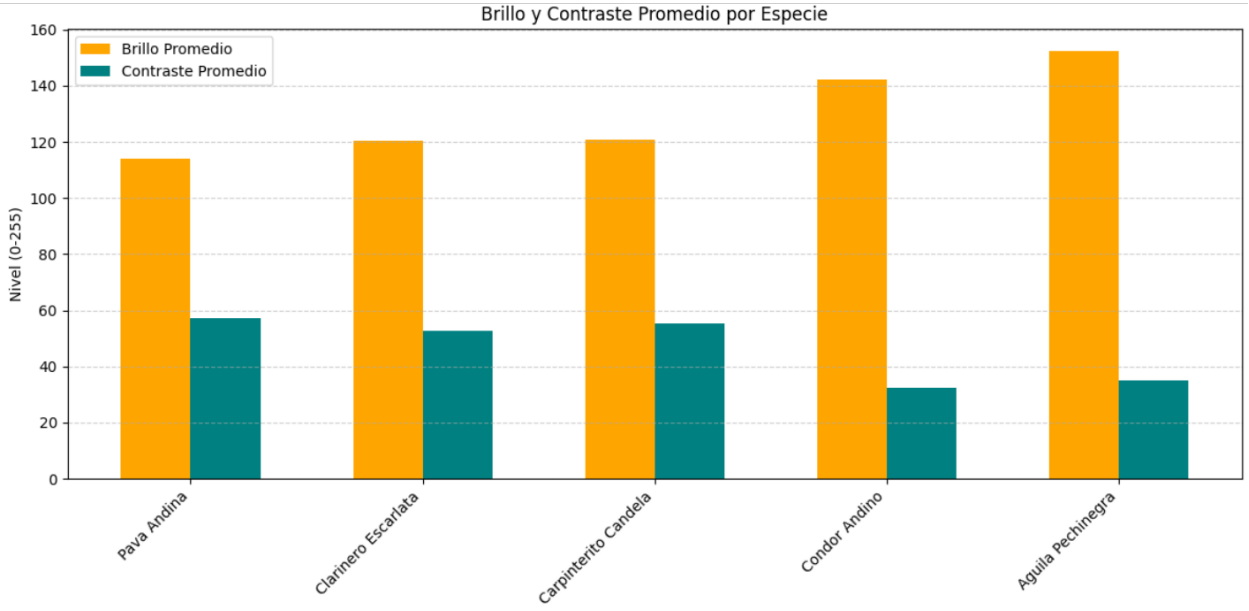


Figura 14: Gráfico de barras de brillo y contraste promedio por especie

Complementando el análisis anterior, se realizó un estudio de la distribución de intensidad de color en una muestra de imágenes, específicamente para la especie Cóndor Andino. Comprender cómo se distribuyen los valores de píxel en las imágenes es fundamental para identificar la presencia de problemas como subexposición, sobreexposición o un bajo contraste general. Para este propósito, se seleccionaron cinco imágenes representativas y se generó el histograma de intensidad de píxeles para cada una. Estos histogramas visualizan la frecuencia con la que aparece cada nivel de brillo (de 0 a 255) en la imagen, proporcionando una idea clara del rango tonal y la concentración de píxeles en ciertas zonas.

En la figura 15, las primeras imágenes muestran que la mayoría de los píxeles se concentraron en los rangos medios y bajos, indicando escenas más oscuras o con menor iluminación general. En contraste, imágenes más cercanas o en vuelo, presentaron una distribución más amplia y una mayor concentración de píxeles en rangos altos de intensidad, lo que sugería una exposición adecuada o la presencia de cielos claros.

Figura 15

Histogramas de intensidad de píxeles para el conjunto Cóndor Andino

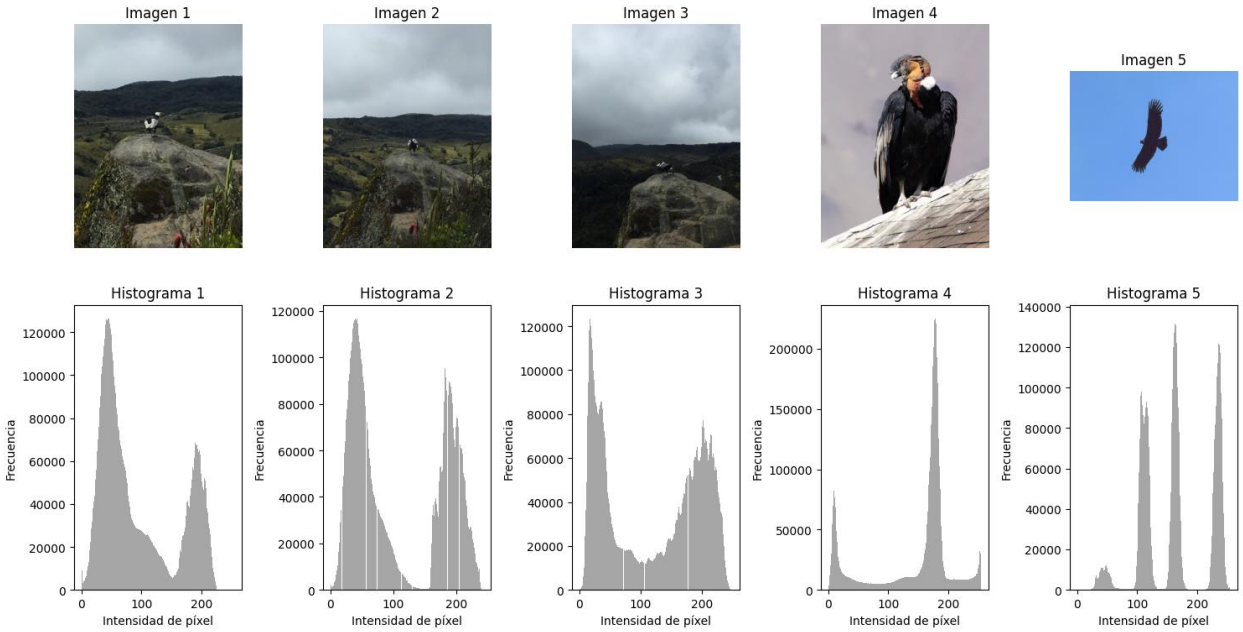


Figura 15: Histogramas de intensidad de píxeles para el conjunto Cóndor Andino

Luego de los análisis de la intensidad, se pudo notar que en todas las carpetas existen fotografías que presentan cambios de color o perdidas de información. Para profundizar en las características internas de las imágenes y evaluar la riqueza textural y la presencia de detalles, se realizó un análisis de la distribución de la variabilidad de píxeles para cada especie. La varianza de los valores de píxel en una imagen sirve como un indicador de la dispersión de los tonos y la uniformidad de la imagen.

El análisis de los histogramas presentes en la Figura 16, proporcionó información valiosa sobre la calidad y diversidad de las imágenes. Se observó que, en la mayoría de las imágenes para todas las especies presentaban una varianza dentro de rangos medios, que se deduce en una buena presencia de detalles y texturas útiles para el reconocimiento. Sin embargo, se identificaron colas en las distribuciones hacia varianzas bajas, indicando la existencia de algunas imágenes con poca información visual en todas las carpetas. Con este análisis final se reafirmó la necesidad de una selección manual para la optimización del conjunto de datos.

Figura 16

Distribución de la variabilidad de píxeles por especie

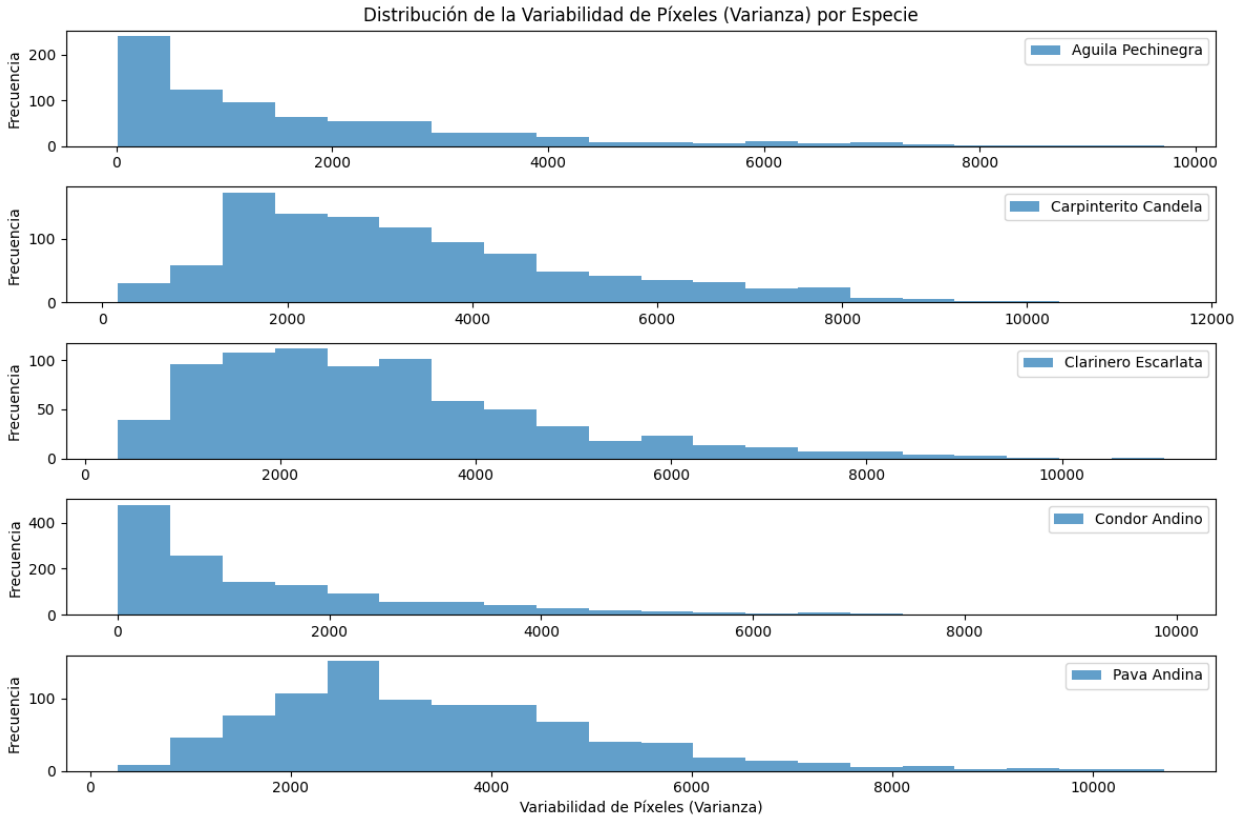


Figura 16: Distribución de la variabilidad de píxeles por especie

4.1.2.3 Implicaciones del Análisis para la Calidad de las Imágenes. El análisis exploratorio y estadístico reveló varias características que impactan directamente en la calidad y la uniformidad de la información disponible para el entrenamiento del modelo de aprendizaje automático. Estos hallazgos, marcaron las pautas necesarias para la estandarización y adecuación de las imágenes. Con el fin de obtener un conjunto de datos robusto, consistente y óptimo para la fase de modelado, las principales implicaciones y acciones requeridas fueron:

- **Presencia de Ruido y Fondos Sobreexpuestos:** Se detectó una alta proporción de imágenes donde el ave no es reconocible debido a la sobreexposición del cielo, la densidad del follaje boscoso o la presencia de otros elementos que distorsionan la percepción del ave. Esto implica la necesidad de una clasificación manual para descartar aquellas imágenes que no contienen el ave de forma clara, o el ruido ambiental predomina sobre la especie, con el fin de mejorar la pureza y la calidad del conjunto de datos de entrenamiento.
- **Diversidad de Modos de Color:** La presencia de imágenes en modos RGB, RGBA y L hizo evidente la necesidad de convertir todas las imágenes a un modo de color uniforme.
- **Variabilidad en Dimensiones:** Se observó una amplia gama de resoluciones y proporciones de aspecto. Esto implica la necesidad de redimensionar y normalizar todas las imágenes a un tamaño fijo y consistente para cumplir con los requisitos de entrada de del modelo y optimizar el uso de recursos computacionales.
- **Desbalance de Clases:** Se identificó una distribución desigual en el número de imágenes por especie. Esto implica la necesidad de implementar estrategias de balanceo de datos como el aumento de datos, para evitar que el modelo presente sesgos hacia las clases mayoritarias y mejorar su rendimiento en la clasificación de especies con menos muestras.

- **Variación en Brillo y Contraste:** Las diferencias encontradas en los niveles de brillo y contraste entre las especies y dentro de las imágenes de una misma especie indican la necesidad de aplicar técnicas de ajuste de contraste. Buscando reducir la sensibilidad del modelo a las condiciones de iluminación y realzar las características visuales distintivas de las aves.

4.1.3 Preparación de los Datos

Una vez completada la fase de exploración y estipulado los necesarios para el preprocesamiento de las imágenes, se procedió a implementar las implicaciones con el fin de transformar el conjunto de imágenes crudas en un formato adecuado y de alta calidad para el entrenamiento del modelo. Buscando estandarizar las características de las imágenes, fue necesario un proceso de selección y filtrado de imágenes, conversión de formatos y modos de color, el redimensionamiento, y la implementación de técnicas de aumento de datos para abordar el desbalance de clases. Todas las técnicas se detallan a continuación en puntos específicos que abarcan una explicación más completa del proceso realizado en esta fase.

4.1.3.1 Filtrado y Selección de Imágenes Relevantes. El primer punto en la preparación fue la selección manual de las fotografías óptimas para el entrenamiento del modelo. Por el análisis previo se concluyó que existía un número considerable de fotografías donde el entorno dominaba la escena, dificultando la visibilidad de la especie. Este ruido visual podía comprometer seriamente la capacidad del modelo para aprender características discriminatorias relevantes.

Para solucionarlo, se implementó una interfaz basada en “ipywidgets” que permitió la revisión individual de cada imagen en el conjunto de datos. Este proceso facilitó la inspección visual de 4823 imágenes y la toma de decisiones para conservar o eliminar fotografías según el

criterio propio. La tarea se tornó extensa, pero necesaria en la búsqueda de un conjunto de imágenes óptimo que garantice la calidad del resultado final. La interfaz, como se muestra en la Figura 17, presentaba una imagen a la vez, junto con botones para su aprobación o descarte, recorriendo cada una de las carpetas.

Figura 17

Interfaz interactiva para la curación manual de imágenes



Figura 17: Interfaz interactiva para la curación manual de imágenes

Terminado el proceso se procedió a recontar las imágenes por especie, otorgando los siguientes resultados:

- Águila Pechinegra: 324 imágenes
- Carpinterito Candela: 413 imágenes
- Clarinero Escarlata: 370 imágenes
- Cóndor Andino: 382 imágenes
- Pava Andina: 366 imágenes

El total de imágenes restantes en el conjunto de datos después del proceso de filtrado manual fue de 1855. Este proceso de eliminación manual resultó en una reducción significativa del tamaño total del conjunto de datos inicial, donde inicialmente se contaba con 4823. Aunque el volumen total disminuyó, se garantizó que las imágenes conservadas presenten al ave de forma clara y con el contexto visual adecuado.

A pesar del filtrado visual, existía la posibilidad de que algunas imágenes estuvieran corruptas, incompletas o en un formato no reconocido por las librerías de procesamiento, lo que podría causar errores durante el entrenamiento del modelo. Para llevar a cabo esta verificación, se recorrió cada imagen restante en el conjunto de datos y se intentó abrirla y verificar su estructura. La función "img.verify()" de PIL permite comprobar si un archivo de imagen es válido sin cargar completamente todos sus datos en memoria, lo que optimiza el proceso.

Figura 18

Implementación del proceso de verificación de archivos de imagen



```
#Verificar la integridad del archivo
try:
    with Image.open(ruta_imagen) as img:
        img.verify()
except (UnidentifiedImageError, OSError):
    # Si ocurre un error de identificación o lectura, la imagen es corrupta
    os.remove(ruta_imagen)
```

Figura 18: Implementación del proceso de verificación de archivos de imagen

Una vez concluida la etapa de eliminación de imágenes no necesarias y aquellas con problemas puntuales, el conjunto final para trabajar la etapa de preprocesamiento estuvo compuesto por 1854 imágenes, distribuidas en 5 carpetas propias de cada especie.

4.1.3.2 Estandarización de Atributos de Imagen. El primer punto a tomar en cuenta en el proceso de estandarización fue el formato de color. Para el primer enfoque del modelo se necesita las imágenes con sus tres canales RGB, por lo tanto, fue necesario cambiar aquellas fotografías con formatos distintos. Esto garantiza que la información de color se pueda procesar de manera consistente en todo el conjunto de datos y evita errores provocados por variabilidad en los canales de color.

Figura 19

Implementación de la conversión de modos de color a RGB

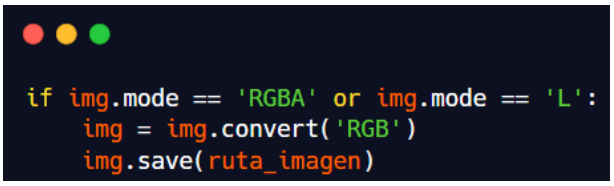


Figura 19: Implementación de la conversión de modos de color a RGB

El siguiente aspecto a solventar fue la variabilidad de dimensiones en las fotografías presentes en el conjunto de datos. Siendo un paso fundamental para satisfacer los requisitos de entrada de la arquitectura de las redes neuronales convolucionales, asegurando una consistencia dimensional en el entrenamiento. El proceso abarcó un redimensionamiento uniforme donde todas las imágenes fueron estandarizadas a un tamaño de 256x256 píxeles.

De esta forma, se aseguró la preservación de un nivel adecuado de detalle visual de las aves, al tiempo que optimiza la eficiencia computacional durante las fases de entrenamiento y validación del modelo. La operación de redimensionamiento se llevó a cabo utilizando la función resize() de la librería Pillow, como se muestra en la Figura 20.

Figura 20

Implementación del proceso de redimensionamiento de imágenes

```python
#Función para redimensionar imágenes
def redimensionar_imagenes(base_dir, size=(256, 256)):
    especies = os.listdir(base_dir)
    for especie in especies:
        carpeta = os.path.join(base_dir, especie)
        imagenes = os.listdir(carpeta)
        for imagen in imagenes:
            ruta_imagen = os.path.join(carpeta, imagen)
            with Image.open(ruta_imagen) as img:
                img_resized = img.resize(size)
                img_resized.save(ruta_imagen)
    print("Proceso de recorte finalizado.")

redimensionar_imagenes('aves_dataset')
```

Figura 20: Implementación del proceso de redimensionamiento de imágenes

Una vez completados los procesos de conversión de modos de color y redimensionamiento, se consideró oportuno realizar una verificación visual de una muestra de imágenes. Para esta comprobación, se seleccionaron cinco imágenes del Clarinero Escarlata y en la Figura 21 se pudo visualizar un formato uniforme.

Figura 21

Muestra de imágenes estandarizadas del Clarinero Escarlata



Figura 21: Muestra de imágenes estandarizadas del Clarinero Escarlata

Para optimizar el rendimiento y la estabilidad del entrenamiento del modelo, se implementó la normalización de los valores de píxel. Este proceso fue fundamental porque escala los valores de intensidad de los píxeles de las imágenes, que originalmente varían en un rango de 0 a 255, a un rango más pequeño y estandarizado, entre 0.0 y 1.0. La normalización evita que los valores de entrada grandes saturen las activaciones de las neuronas en las primeras capas del modelo y acelera la convergencia durante el entrenamiento al situar todos los atributos en una escala similar.

Una ventaja en este paso de normalización fue que se realizó de forma dinámica y en tiempo real durante la carga de las imágenes para el entrenamiento. Para ello, se utilizó la clase "ImageDataGenerator", que no solo facilita la normalización a través del argumento rescale=1./255, sino que también suprimió la necesidad de cargar y almacenar la totalidad del conjunto de datos de imágenes en arreglos NumPy en memoria RAM. Cada generador de imágenes fue configurado para leer desde el directorio base, identificando automáticamente las diferentes especies como clases y proporcionando los datos ya normalizados y con sus etiquetas en formato one-hot encoded listos para el procesamiento por el modelo.

Figura 22

Muestra de imágenes de las cinco especies normalizadas

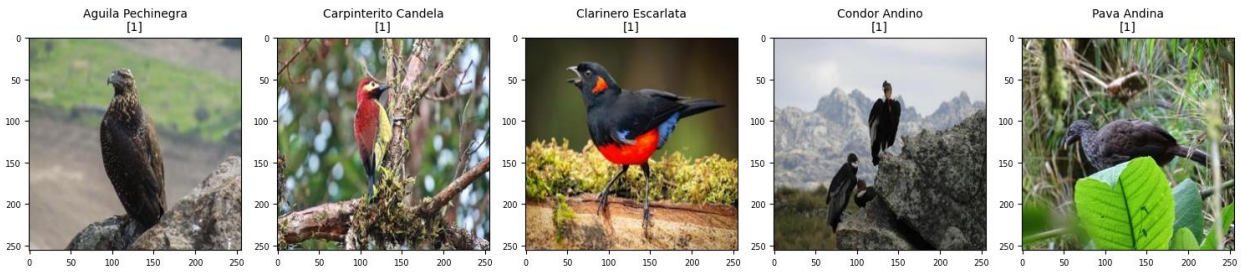


Figura 22: Muestra de imágenes de las cinco especies normalizadas

La siguiente consideración para mejorar la calidad de las imágenes fue la aplicación de filtros de suavizado con el fin de reducir el ruido visual y potenciar las características de las aves. Se experimentó con la aplicación de un filtro Gaussiano, que es comúnmente utilizado para suavizar imágenes y eliminar ruido de alta frecuencia. La idea era atenuar los detalles irrelevantes del fondo para mejorar el enfoque del ave, pero luego de analizar los resultados en dos fotografías del Cóndor Andino, las métricas cuantitativas del Error Cuadrático Medio (MSE) y la Relación Pico-Señal a Ruido (PSNR) revelaron que, aunque el ruido se reducía, el filtro Gaussiano también difuminaba significativamente los detalles finos esenciales de las plumas y contornos del ave.

Esta pérdida de detalle crítico fue considerada perjudicial para el reconocimiento de patrones por parte del modelo, ya que el modelo depende en gran medida de las texturas y bordes específicos para la clasificación de especies. Por lo tanto, se concluyó que la aplicación de filtros de suavizado como el Gaussiano no era beneficiosa para este conjunto de datos, ya que comprometía la riqueza de los detalles necesarios para una clasificación precisa.

Figura 23

Comparación visual de imagen original y con filtro Gaussiano aplicado



Figura 23: Comparación visual de imagen original y con filtro Gaussiano aplicado

4.1.3.3 Data Augmentation. Finalizado el proceso de eliminación manual en pasos anteriores, la principal consecuencia de esa acción fue la considerable reducción de imágenes para cada especie y la necesidad de emplear una solución que evite la insuficiencia de datos para entrenar el modelo. Se aplicó la técnica de aumento de datos (Data Augmentation), la cual es crucial en conjuntos de datos de tamaño limitado, ya que permite expandir artificialmente la diversidad del dataset y ayuda al modelo a ser menos sensible a variaciones en la posición, orientación, escala o condiciones de iluminación de las aves, lo que reduce el riesgo de overfitting y mejora su rendimiento en datos no vistos.

El proceso consistió en generar nuevas versiones de las imágenes existentes aplicando transformaciones geométricas y de color de forma aleatoria. Se utilizaron las siguientes operaciones de aumento, implementadas a través de "ImageDataGenerator" de Keras, y se generaron tres imágenes aumentadas por cada imagen original:

- **rotation_range=25:** Rotaciones aleatorias de hasta 25 grados.
- **width_shift_range=0.1:** Desplazamientos horizontales aleatorios de hasta el 10% del ancho de la imagen.
- **height_shift_range=0.1:** Desplazamientos verticales aleatorios de hasta el 10% del alto de la imagen.
- **zoom_range=0.15:** Zooms aleatorios de hasta el 15%.
- **horizontal_flip=True:** Volteo horizontal aleatorio de las imágenes.
- **brightness_range=(0.8, 1.2):** Ajustes aleatorios del brillo entre el 80% y el 120% del valor original.

Figura 24

Muestra de imágenes generadas por aumento de datos del conjunto Pava Andina



Figura 24: Muestra de imágenes generadas por aumento de datos del conjunto Pava Andina

4.2 Construcción del Modelo de Clasificación

La siguiente sección se define como la parte principal en el desarrollo del proyecto, en la cual se detalla el proceso de construcción del modelo de clasificación, en busca de un resultado que permita la exitosa categorización automática de especies. El desarrollo de la arquitectura de la red neuronal se abordó considerando la naturaleza de los datos de imagen, que comprenden tanto representaciones a color (RGB) como, el ajuste de parámetros para un enfoque en escala de grises que permita explorar el impacto de los canales cromáticos. El proceso abarcó desde la definición de la topología de la red hasta la preparación de los datos en conjuntos de entrenamiento y validación, cruciales para el aprendizaje supervisado. Se presentarán los componentes arquitectónicos seleccionados y las estrategias implementadas para la configuración del modelo.

4.2.1 Modelado

En esta etapa del marco metodológico, se procede a la construcción formal de las arquitecturas de red neuronal convolucional (CNN) que serán entrenadas para reconocer y clasificar las diferentes especies seleccionadas. Esta etapa implicó la definición de la topología, la selección de capas, funciones de activación y estrategias de optimización, adaptando la

complejidad del modelo a la naturaleza y volumen del conjunto de datos preprocesado. Se explorarán y detallarán los procesos y configuraciones específicas de los enfoques elegidos en busca del mejor resultado posible.

4.2.1.1 Arquitectura Red Neuronal Convolucional – RGB. Para el primer modelo, se diseñó e implementó una arquitectura de Red Neuronal Convolucional (CNN) personalizada. La configuración está definida para aceptar imágenes de 256×256 píxeles con tres canales de color (RGB), reflejando las dimensiones y el formato estandarizados en la etapa de preprocesamiento.

La arquitectura propuesta consta de una serie de bloques convolucionales seguidos por capas de pooling, culminando en capas densamente conectadas para la clasificación final. Específicamente, los componentes se detallan de la siguiente manera:

- **Capas Convolucionales (Conv2D):** Se utilizaron tres capas convolucionales sucesivas, con un aumento progresivo en el número de filtros, 32, 64 y 128, y un tamaño de kernel de 3×3 en todas ellas. La función de activación ReLU se aplicó después de cada convolución. Estas capas son responsables de aprender características locales de la imagen, como bordes, texturas y patrones, a diferentes niveles de abstracción.

- **Capas de Max Pooling (MaxPooling2D):** Tras cada capa convolucional, se incorporó una capa de Max Pooling de tamaño 2×2. Estas capas reducen la dimensionalidad espacial de los mapas de características, lo que contribuye a disminuir el número de parámetros, controlar el overfitting y hacer el modelo más robusto a pequeñas traslaciones o deformaciones en la entrada.

- **Capa de Aplanamiento (Flatten):** Después de las capas convolucionales y de pooling, se añadió una capa Flatten para transformar la salida multidimensional de las capas convolucionales en un vector unidimensional. Esto fue necesario para conectar las características extraídas a las capas densamente conectadas.

- **Capa de Abandono (Dropout):** Se incluyó una capa Dropout con una tasa del 0.5. Esta técnica de regularización desactiva aleatoriamente un porcentaje de neuronas durante el entrenamiento, lo que ayuda a prevenir el overfitting al obligar a la red a aprender representaciones más robustas y menos dependientes de neuronas específicas.

- **Capas Densas (Dense):** El modelo culmina con dos capas densamente conectadas. La primera capa cuenta con 128 neuronas y utiliza la activación ReLU, sirviendo como una capa de características de alto nivel. La capa final es una capa densa de salida con 5 neuronas y la función de activación Softmax. La cual es ideal para problemas de clasificación multiclase, ya que produce una distribución de probabilidades sobre las clases, indicando la confianza del modelo en cada predicción.

El modelo fue compilado utilizando el optimizador Adam, que es un algoritmo de optimización adaptativo popular y eficiente para el entrenamiento de redes neuronales. La función de pérdida seleccionada fue “categorical_crossentropy”, adecuada para problemas de clasificación multiclase con etiquetas one-hot encoded. La métrica de evaluación principal para monitorear el rendimiento durante el entrenamiento fue la “accuracy”. En la Tabla 4 se detalla el resumen de la

arquitectura de la red. Además, luego de ver el resultado, fue interesante entender ciertos parámetros resultantes de la construcción:

- **Total params: 14,839,621 (56.61 MB):** Número total de pesos y sesgos que el modelo debe aprender, reflejando su complejidad.
- **Trainable params: 14,839,621 (56.61 MB):** Indica que todos los parámetros del modelo se ajustarán durante el entrenamiento.
- **Non-trainable params: 0 (0.00 B):** Muestra que no hay parámetros fijos, lo cual ocurre cuando no se utilizan capas pre-entrenadas.

Tabla 4

Arquitectura y Resumen de Capas del Modelo CNN-RGB

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dropout (Dropout)	(None, 115200)	0
dense (Dense)	(None, 128)	14,745,728
dense_1 (Dense)	(None, 5)	645

Tabla 4: Arquitectura y Resumen de Capas del Modelo CNN-RGB

Una vez definida la arquitectura del modelo, el siguiente paso fue la preparación de los conjuntos de datos que alimentaran la red neuronal durante el entrenamiento y la validación. Para

gestionar eficientemente este proceso y aprovechar las capacidades de preprocesamiento, se utilizaron los generadores de imágenes “ImageDataGenerator” de Keras.

Se configuraron dos generadores distintos para crear los flujos de datos de entrenamiento y validación:

- **Generador para Entrenamiento (train_datagen):** Encargado de proveer los datos al modelo durante la fase de aprendizaje. El argumento validation_split=0.2 se utilizó para reservar un 20% del total de imágenes para el conjunto de validación y empezar el entrenamiento con 80%.

- **Generador para Validación (val_datagen):** Exclusivamente para la fase de validación., donde se configuró con validation_split=0.2 para asegurar que procesara la misma porción de datos reservada para validación por el train_datagen.

Ambos generadores utilizaron el método “flow_from_directory()” para cargar imágenes directamente desde las carpetas de las especies. Se especificó un batch_size=32 para el procesamiento en lotes y el argumento “subset” fue clave para que cada generador accediera a la porción correcta del dataset dividido.

La división de los datos resultó con 5715 imágenes para el conjunto de entrenamiento y 1426 imágenes para el conjunto de validación, distribuidas en las 5 clases de especies de aves.

Figura 25

Configuración de ImageDataGenerator para la división de datos

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os

base_dir = 'aves_dataset'
img_size = (256, 256)
batch_size = 32
especies = [
    "Aguila_Pechinegra",
    "Carpinterito_Candela",
    "Clarinero_Escarlata",
    "Condor_Andino",
    "Pava_Andina"
]

# Generador para entrenamiento con normalización y aumento
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,   # 20% validación
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Generador para validación con solo normalización
val_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

# Flujo de entrenamiento
train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=img_size,
    classes=especies,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

# Flujo de validación
val_generator = val_datagen.flow_from_directory(
    base_dir,
    target_size=img_size,
    classes=especies,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    shuffle=False
)
```

Figura 25: Configuración de ImageDataGenerator para la división de datos

TensorFlow tiene una herramienta que permite la visualización del comportamiento del modelo en términos de precisión y perdida, con una configuración de un archivo que guarde las secuencias del entrenamiento. Por lo tanto, se configuró un sistema de registro (logs) para monitorear el progreso y rendimiento a lo largo de las épocas. Esto se realizó mediante la integración de un callback de TensorBoard.

Una vez establecidos los logs, se procedió al entrenamiento del modelo CNN-RGB. El entrenamiento se llevó a cabo utilizando los generadores de datos previamente configurados, que alimentan al modelo con lotes de imágenes preprocesadas y aumentadas. El proceso de entrenamiento, que abarca 30 épocas, se detalla en la Figura 26. El proceso fue realizado con la capacidad de computo propia, CPU Ryzen 7, GTX 1650Ti y 16GB de memoria RAM. Tomando alrededor de 45 minutos para completar el entrenamiento.

Figura 26

Configuración y Ejecución del Entrenamiento del Modelo CNN

```
# Entrenamiento del modelo
history = modelo.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size,
    epochs=30,
    callbacks=[tensorboard_callback]
)
```

Figura 26: Configuración y Ejecución del Entrenamiento del Modelo CNN

4.2.1.2 Arquitectura Red Neuronal Convolucional – Escala de Grises. El segundo enfoque implementado en el desarrollo del sistema fue la escala de grises. El objetivo es analizar con esta arquitectura la influencia de la información cromática en la capacidad de clasificación del modelo. Esta arquitectura es esencialmente idéntica a la diseñada para imágenes RGB, con la única diferencia en la capa de entrada, que ahora está configurada para aceptar imágenes de 256 píxeles con un único canal de intensidad. Esta modificación resultó en una ligera reducción en el número de parámetros de la primera capa convolucional, dada la menor cantidad de canales de entrada que procesar.

El proceso de transformación de las imágenes a escala de grises se realizó de manera sistemática. Se iteró sobre cada imagen presente en el dataset original y se convirtió a un formato de escala de grises. Las imágenes transformadas se guardaron en un nuevo directorio para mantener la estructura de carpetas por especie.

Figura 27

Muestra de imágenes transformadas a escala de grises para cada especie.



Figura 27: Muestra de imágenes transformadas a escala de grises para cada especie

Para la construcción del modelo CNN en escala de grises, se optó por replicar la arquitectura fundamental del modelo desarrollado para imágenes RGB, con el fin de establecer una base de comparación directa y evaluar la importancia de la información cromática en la tarea

de clasificación. Esta estrategia permite atribuir las diferencias en el rendimiento principalmente a la naturaleza de los datos de entrada y no a variaciones en la complejidad o diseño de la red.

La única modificación sustancial en la arquitectura de la red reside en la capa de entrada. El modelo RGB fue configurado para recibir imágenes con tres canales de color (RGB) (256, 256, 3), caso contrario, para el modelo de escala de grises se debe procesar imágenes con un único canal de intensidad, (256, 256, 1). Este cambio impacta directamente en la primera capa convolucional, donde el cálculo de parámetros para esta capa se ajusta para reflejar un solo canal de entrada, resultando en 320 parámetros, en contraste con los 896 parámetros del modelo RGB para la misma capa.

La arquitectura complementaria, incluyendo el número de capas convolucionales subsiguientes, las capas de Max Pooling, Flatten, Dropout y las capas densamente conectadas, se mantuvo idéntica a la configuración del modelo RGB.

Tabla 5

Arquitectura y Resumen de Capas del Modelo CNN-Escala de Grises

Capa (Tipo)	Forma de Salida	Parámetros
conv2d_6 (Conv2D)	(None, 254, 254, 32)	320
max_pooling2d_6 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_7 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_7 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_8 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_8 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten_2 (Flatten)	(None, 115200)	0
dropout_2 (Dropout)	(None, 115200)	0
dense_4 (Dense)	(None, 128)	14,745,728
dense_5 (Dense)	(None, 5)	645

Tabla 5: Arquitectura y Resumen de Capas del Modelo CNN-Escala de Grises

El entrenamiento del modelo con escala de grises fue similar al modelo RGB, se replicaron los procedimientos establecidos y los ajustes específicos para el formato de entrada de las imágenes. Los generadores fueron configurados para la normalización de píxeles y una división del 80% para entrenamiento y 20% para validación. Asimismo, se configuró los registros para el monitoreo con TensorBoard, lo que permitirá posteriormente comparar los resultados en la etapa de evaluación. Finalmente, el modelo en escala de grises fue compilado y entrenado, al igual que el proceso pasado, con la capacidad de computo local, se llegó a completar el proceso en 42 minutos aproximadamente.

Figura 28

Detalle de la época final del Entrenamiento del Modelo CNN-Escala de Grises

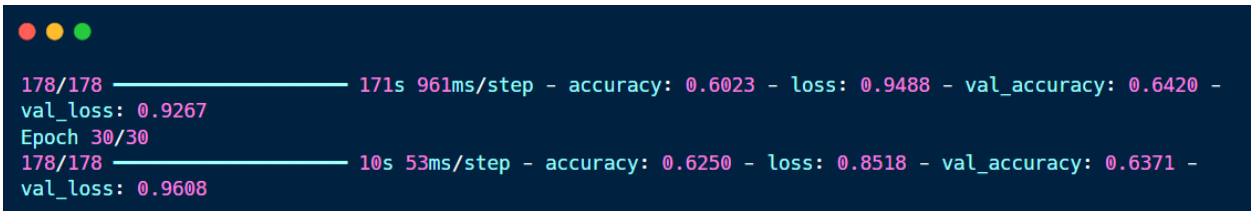


Figura 28: Detalle de la época final del Entrenamiento del Modelo CNN-Escala de Grises

4.3 Evaluación del Desempeño del Modelo de Clasificación

Una vez finalizado el proceso de entrenamiento de los modelos CNN en su versión RGB y escala de grises, el siguiente paso fundamental consiste en evaluar su desempeño. Esta fase es fundamental dentro del marco de trabajo porque permite cuantificar la capacidad de cada modelo para generalizar y realizar predicciones precisas sobre datos no vistos. La evaluación se llevó a cabo utilizando un conjunto de métricas de desempeño estándar en problemas de clasificación, las cuales permiten analizar la precisión, robustez y el balance entre sensibilidad y especificidad de las predicciones de los modelos.

4.3.1 Evaluación

El objetivo principal de la penúltima fase es proporcionar una visión clara y objetiva de los resultados obtenidos y la efectividad en la tarea de identificación de las cinco especies seleccionadas. El punto de partida es la comparación de métricas entre ambos procesos de entrenamiento y validación.

4.3.1.1 Visualización del Rendimiento del Modelo – RGB. La Figura 29 muestra la evolución de la precisión del modelo RGB durante el entrenamiento, la línea roja representa el conjunto de entrenamiento y la verde el de validación. Se observó una clara tendencia ascendente en ambas curvas a lo largo de las 30 épocas, lo que indicó que el modelo estaba aprendiendo y mejorando su capacidad de clasificación. Las primeras épocas mostraron un crecimiento rápido y fluctuaciones, a partir de la época 15, la mejora se tornó más gradual y las curvas tendieron a estabilizarse, lo que sugirió una convergencia del entrenamiento.

Al finalizar, la precisión de entrenamiento alcanzó un 90.63%, mientras que la precisión de validación se situó alrededor del 83.74%, lo que proporciona una medida final del desempeño generalizable del modelo.

Figura 29

Curvas de Precisión del Modelo CNN-RGB durante el Entrenamiento

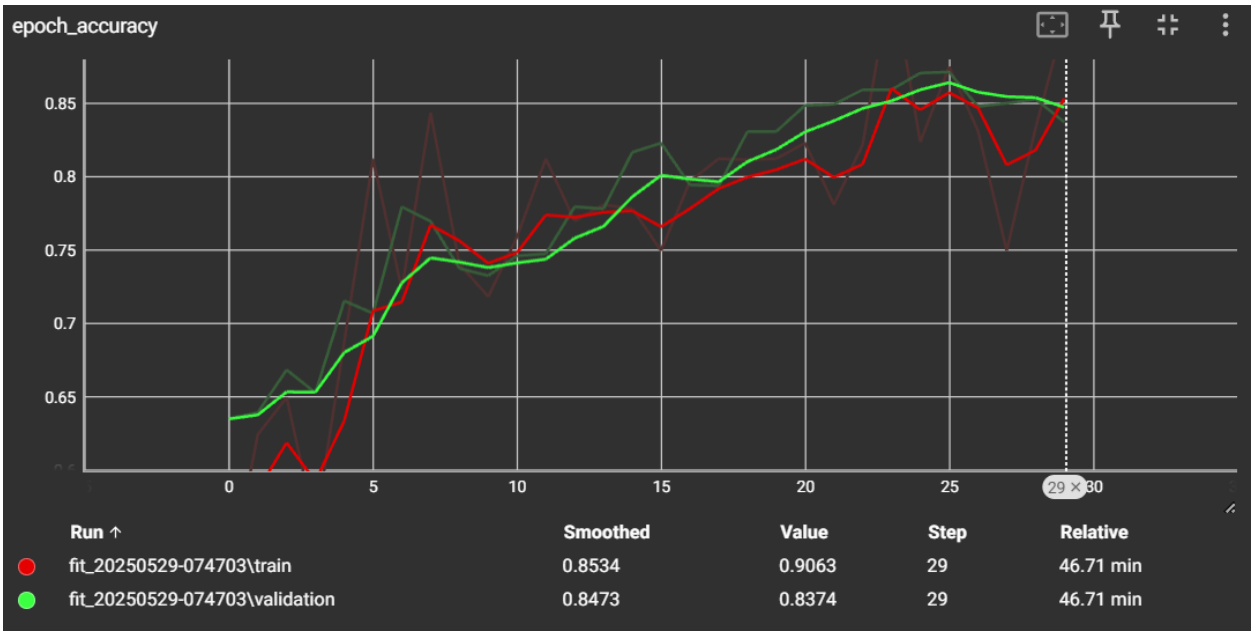


Figura 29: Curvas de Precisión del Modelo CNN-RGB durante el Entrenamiento

Complementando el primer gráfico, también fue importante el análisis del gráfico de perdida durante el entrenamiento. En la Figura 30 se muestra como ambas curvas de pérdida mostraron una clara tendencia decreciente a lo largo de las 30 épocas. Esto indicó que el modelo estaba reduciendo su error en las predicciones, aprendiendo eficazmente a ajustar sus parámetros para acercar sus resultados a las etiquetas verdaderas.

Además, la divergencia entre ambas curvas de pérdida, donde el error en datos no vistos fue mayor, reafirmó la presencia de un sobreajuste moderado. A pesar de esto, el modelo mantuvo una capacidad de generalización aceptable. Al finalizar, la pérdida de entrenamiento fue de 0.2511 y la de validación de 0.3985.

Figura 30

Curvas de Pérdida del Modelo CNN-RGB durante el Entrenamiento

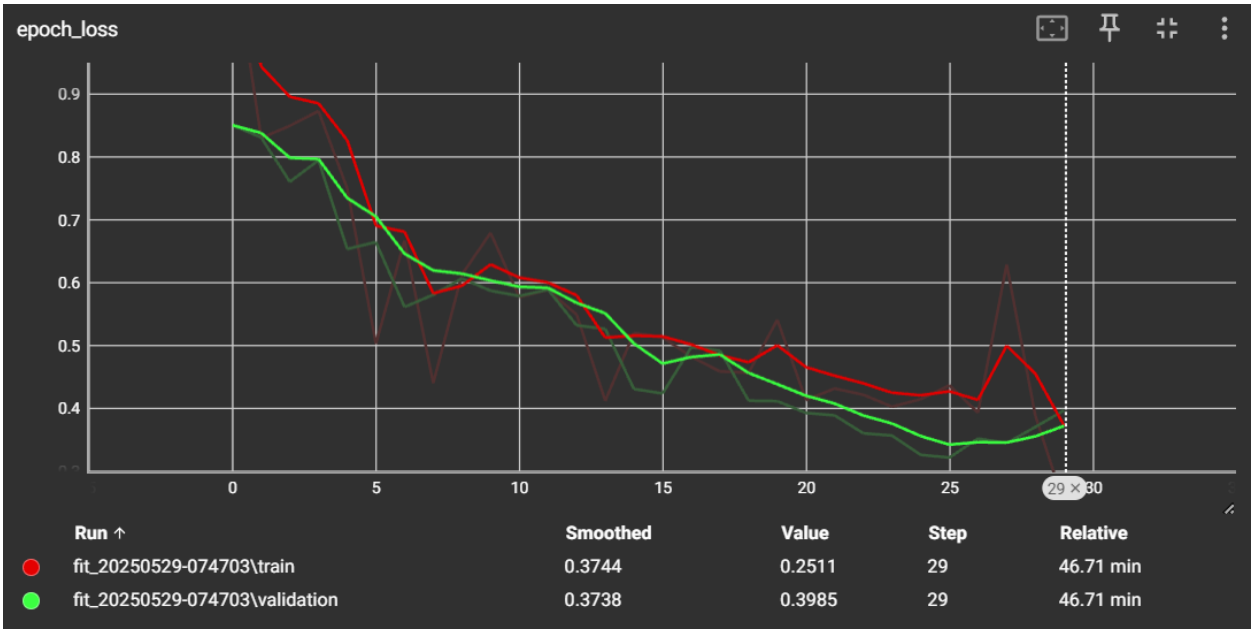


Figura 30: Curvas de Pérdida del Modelo CNN-RGB durante el Entrenamiento

4.3.1.2 Visualización del Rendimiento del Modelo – Escala de Grises. La evaluación del modelo en escala de grises siguió un proceso similar al del modelo RGB. En la Figura 31 se muestra la evolución de la precisión para los conjuntos de entrenamiento y validación de este modelo, en donde se observó que la precisión de ambos conjuntos mostró una tendencia general de mejora a lo largo de las 30 épocas. Las primeras épocas se caracterizaron por un ascenso rápido y fluctuaciones significativas, finalizando en las últimas épocas con curvas que tienden a estabilizarse, indicando una convergencia. La precisión de validación fue de 63.71%, ligeramente superior a la precisión de entrenamiento de 62.50%. Sin embargo, el nivel general de precisión alcanzado fue significativamente menor que el del modelo RGB.

Figura 31

Curvas de Precisión del Modelo CNN-Escala de Grises durante el Entrenamiento.

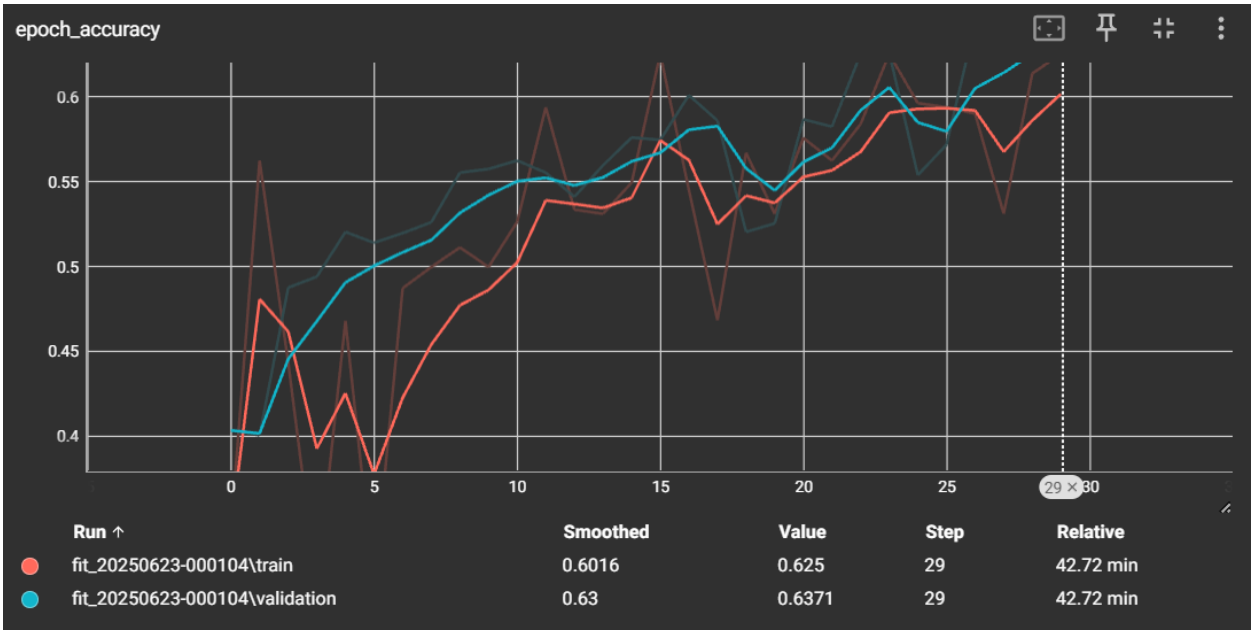


Figura 31: Curvas de Precisión del Modelo CNN-Escala de Grises durante el Entrenamiento

Finalmente, el gráfico de perdida para el modelo con escala de grises presentó curvas que exhibieron una clara tendencia decreciente a lo largo de las 30 épocas, lo que confirmó que el modelo estaba reduciendo su error a medida que avanzaba el entrenamiento.

No obstante, la pérdida de validación se mantuvo consistentemente por encima de la pérdida de entrenamiento, denotando una mayor fluctuación en la curva de validación. A pesar de esto, la pérdida de validación continuó una tendencia general a la baja, lo que concluye que, si bien el modelo se ajustó mejor a los datos de entrenamiento, mantuvo una capacidad de generalización aceptable. Al finalizar las 30 épocas, la pérdida de entrenamiento fue de 0.8518 y la pérdida de validación de 0.9608.

Figura 32

Curvas de Pérdida del Modelo CNN-Escala de Grises durante el Entrenamiento.

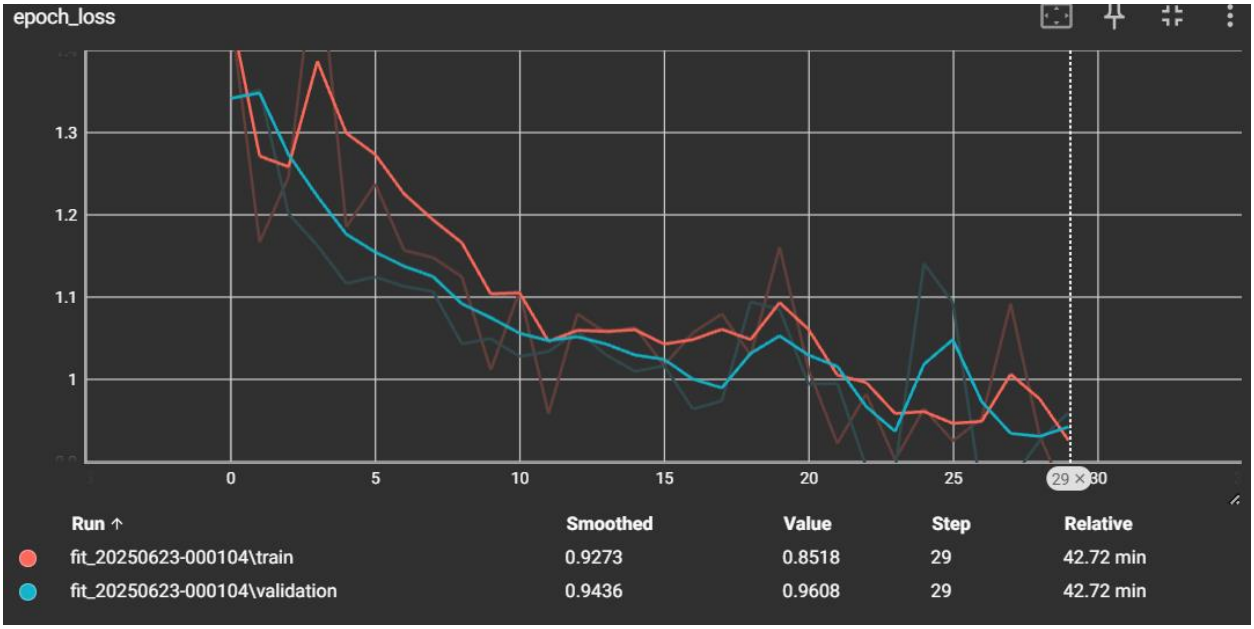


Figura 32: Curvas de Pérdida del Modelo CNN-Escala de Grises durante el Entrenamiento

4.3.1.3 Demostración de la Capacidad Predictiva del Modelo. Para probar la capacidad de predicción de los modelos entrenados fue necesaria su exportación y persistencia, proceso que permite guardar el estado completo de un modelo en un formato que puede ser cargado y utilizado en cualquier momento sin necesidad de reentrenarlo. Para el proyecto, el formato elegido fue HDF5 (.h5), un estándar para el almacenamiento de grandes volúmenes de datos numéricos, comúnmente utilizado en el ecosistema de TensorFlow/Keras para la serialización de modelos de Deep Learning.

La prueba de su capacidad predictiva se planificó utilizando un conjunto de imágenes que nunca fueron vistas por los modelos durante las fases de entrenamiento y validación. Buscando obtener una medida imparcial y realista de la generalización del modelo, demostrando su aptitud para clasificar nuevas aves en un escenario práctico.

Las predicciones se hicieron de forma similar para ambos casos, las imágenes de prueba fueron preparadas, redimensionadas y ajustadas para que el modelo pudiera procesarlas correctamente. El proceso fue diseñado para revisar todas las fotos en una carpeta de prueba específica y realizar los ajustes necesarios de entrada que permita el modelo dependiendo el enfoque. La respuesta se tradujo al nombre del ave, permitiendo la prueba de las predicciones con cualquier foto nueva, siempre que sea de un tipo de ave que el modelo ya conoce, y así mostrar claramente su clasificación. En la Figura 33 se muestra la especie de la foto original y su clase predicha por el modelo.

Figura 33

Predicciones del Modelo CNN-RGB sobre imágenes nuevas

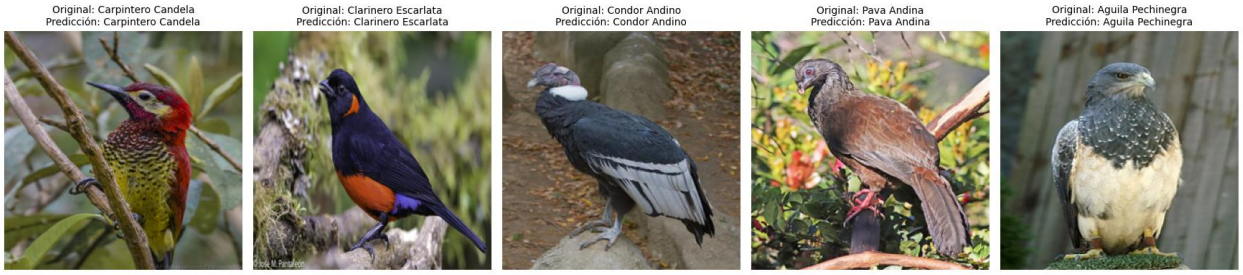


Figura 33: Predicciones del Modelo CNN-RGB sobre imágenes nuevas

El modelo en escala de grises siguió un proceso similar: las imágenes de prueba fueron convertidas a blanco y negro, redimensionadas y preparadas para su entrada al modelo. Dado que este modelo obtuvo una precisión menor, es notorio los fallos en la clasificación de ciertas especies, ya que la predicción en este caso se hace basándose únicamente en la información de luminosidad.

Figura 34

Predicciones del Modelo CNN-Escala de Grises sobre imágenes nuevas

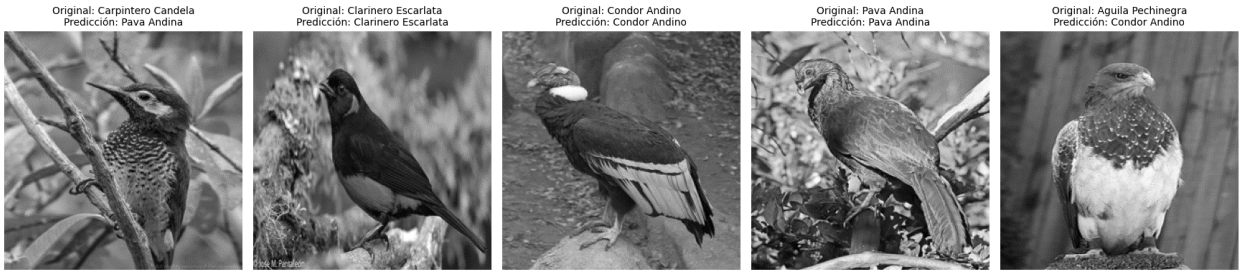


Figura 34: Predicciones del Modelo CNN-Escala de Grises sobre imágenes nuevas

4.3.1.4 Análisis de las Métricas de Evaluación. Con el objetivo comprender a profundidad como se desempeñaron los modelos fue indispensable las evaluaciones cuantitativas utilizando métricas estándar en clasificación, complementadas con la visualización de la matriz de confusión. Estas métricas detallan las identificaciones correctamente o incorrectamente clasificadas para cada especie utilizada para el proyecto.

El modelo RGB fue aquel con la exactitud general más alta con 83.6%, lo que indica una buena capacidad para clasificar correctamente la mayoría de las imágenes. Luego de observar las métricas por especie presentadas en la Tabla 6, se identifican puntos fuertes y áreas con margen de mejora que son importantes detallar de la siguiente forma:

- **Precision:** La especie Clarinero Escarlata exhibió la precisión más alta con 96.7%, lo que significa que las predicciones positivas para esta clase son muy fiables. En contraste, con el Cóndor Andino que presentó la precisión más baja con 80.7%, lo que sugiere que el modelo tuvo algunas dificultades para evitar falsas identificaciones para esta especie. Especies como Águila Pechinegra, Carpintero Candela y Pava Andina también mostraron una buena precisión, todas por encima del 88%.

- **Recall:** La especie Carpintero Candela exhibió la exhaustividad más alta con 93.9%, indicando que el modelo fue muy efectivo en identificar la gran mayoría de las instancias reales de esta ave. Sin embargo, el Cóndor Andino registró la exhaustividad más baja con 73.6%, lo que implica que el modelo pasó por alto una parte de los valores reales.

- **F1-Score:** La especie Carpinterito Candela logró el puntaje más alto con 90.9%, reflejando un excelente equilibrio entre precisión y exhaustividad para esta especie y posicionándola como la mejor clasificada. La especie con más problemas fue el Cóndor Andino con 76.9%, lo cual reitera que esta clase fue la más desafiante para el modelo.

Tabla 6

Métricas de Evaluación para el Modelo CNN-RGB

Especie	Precision	Recall	F1-Score	Soporte
Águila Pechinegra	0.67	0.83	0.74	250
Carpinterito Candela	0.88	0.94	0.91	314
Clarinero Escarlata	0.97	0.82	0.89	287
Cóndor Andino	0.81	0.74	0.77	295
Pava Andina	0.89	0.85	0.87	280
macro avg	0.84	0.84	0.84	1426
weighted avg	0.85	0.84	0.84	1426
Accuracy	0.84	-	-	-

Tabla 6: Métricas de Evaluación para el Modelo CNN-RGB

Es importante mencionar que se complementó el análisis con la visualización de la matriz de confusión, presente en la Figura 35. Con la cual se obtuvo una visualización detallada de cómo el modelo clasificó cada especie, mostrando las predicciones correctas y los errores de clasificación. Detallando los resultados, obtenemos las siguientes conclusiones de la matriz:

- **Águila Pechinegra:** Fue clasificada correctamente en 207 ocasiones. Sin embargo, 39 veces fue predicha como Cóndor Andino y 3 veces como Pava Andina.
- **Carpintero Candela:** Se identificó correctamente 295 veces. Fue confundida 3 veces con Águila Pechinegra y 14 veces con Pava Andina.
- **Clarinero Escarlata:** Fue clasificada correctamente en 236 ocasiones. Las confusiones incluyeron 7 veces como Águila Pechinegra y 33 veces como Carpinterito Candela.
- **Cóndor Andino:** Se identificó correctamente 217 veces. Pero fue clasificado erróneamente en 76 ocasiones como Águila Pechinegra.
- **Pava Andina:** Fue clasificada correctamente en 238 ocasiones. Sus confusiones incluyeron 18 veces como Águila Pechinegra, 7 veces como Carpinterito Candela, 7 veces como Clarinero Escarlata y 10 veces como Cóndor Andino.

Una vez concluido el análisis, se confirmó que las especies Clarinero Escarlata y Carpintero Candela fueron las mejor clasificadas por el modelo RGB, mostrando una alta fiabilidad en sus predicciones y una excelente capacidad para identificar sus respectivas instancias. Por otro lado, la especie Cóndor Andino mostró mayor confusión en el modelo, siendo frecuentemente mal clasificada como Águila Pechinegra, dando a entender que, en su conjunto de imágenes respectiva, donde el ave fue fotografiada en cielos despejados pueda ser la causante de asociar características similares para ambas especies. Aun así, el proceso de clasificación para cada especie está bajo un margen óptimo de rendimiento.

Figura 35

Matriz de Confusión del Modelo RGB

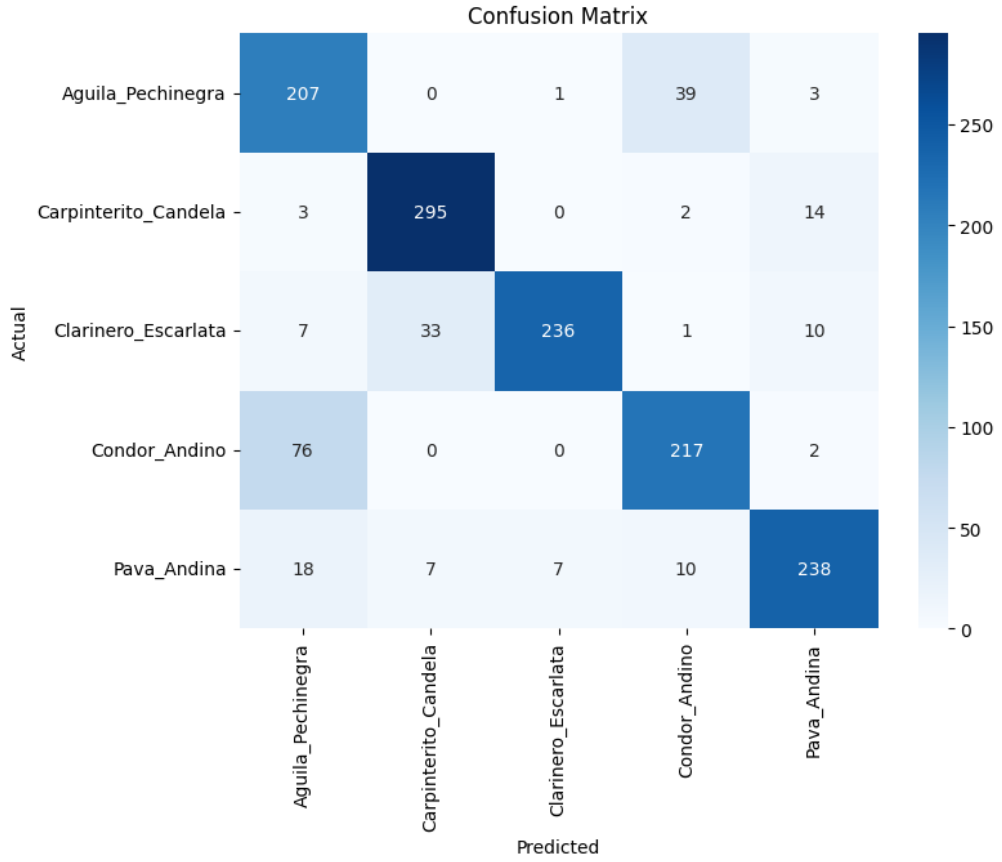


Figura 35: Matriz de Confusión del Modelo RGB

De la misma manera, para el análisis del modelo en escala de grises es importante desglosar los componentes que devolvieron las métricas y entender los puntos altos. En principio el modelo demostró una exactitud general del 64%, lo que indica una capacidad de clasificación correcta moderada, pero significativamente menor en comparación con el modelo RGB. Luego de observar las métricas por especie presentadas en la Tabla 6, se identificaron puntos aspectos claves y áreas con margen de mejora que son importantes detallar de la siguiente forma:

- **Precisión:** La especie Clarinero Escarlata exhibió la precisión más alta con 77%, un resultado relativamente fiable. En contraste, la especie Pava Andina presentó la precisión más baja con 57%, sugiriendo que el modelo tuvo dificultades significativas para evitar falsas identificaciones para esta especie. Las especies Águila Pechinegra, Carpintero Candela" y Cóndor Andino tuvieron precisiones intermedias, entre el 61% y el 66%.

- **Recall:** La especie Pava Andina mostró la exhaustividad más alta con 75%, seguida de cerca por el Cóndor Andino con 73%, indicando que el modelo fue más efectivo en identificar la mayoría de las instancias reales de estas aves. Sin embargo, el "Clarinero Escarlata" registró la exhaustividad más baja con un notable 45%, lo que implica que el modelo pasó por alto y los clasificó en su mayoría incorrectamente.

- **F1-Score:** La especie Cóndor Andino logró el puntaje más alto con 66%, compartiendo valor con el Águila Pechinegra y Pava Andina, ambas con 65%, lo que refleja un equilibrio relativamente mejor entre precisión y exhaustividad para estas clases dentro del modelo en escala de grises. La especie con más problemas fue el Clarinero Escarlata con 57%, lo cual reitera que esta clase fue la más complicada para el modelo al considerar tanto sus predicciones correctas como su capacidad para encontrar todas las instancias.

Tabla 7

Métricas de Evaluación para el Modelo CNN-Escala de Grises

Especie	Precision	Recall	F1-Score	Soporte
Aguila_Pechinegra	0.65	0.64	0.65	250
Carpinterito_Candela	0.66	0.63	0.64	314
Clarinero_Escarlata	0.77	0.45	0.57	287
Condor_Andino	0.61	0.73	0.66	295
Pava_Andina	0.57	0.75	0.65	280
macro avg	0.65	0.64	0.63	1426
weighted avg	0.65	0.64	0.63	1426
Accuracy	0.64	-	-	-

Tabla 7: Métricas de Evaluación para el Modelo CNN-Escala de Grises

En la Figura 36 se detalla la matriz de confusión para el modelo con escala de grises. Con ella se obtuvo una visualización detallada de cómo el modelo clasificó cada especie, mostrando las predicciones correctas y los errores de clasificación. Profundizando en los resultados otorgados por la matriz se puede desglosar los siguientes puntos importantes:

- **Águila Pechinegra:** Fue clasificada correctamente en 161 ocasiones. Sin embargo, fue predicha erróneamente 11 veces como Carpintero Candela, 7 veces como Clarinero Escarlata, 62 veces como Cóndor Andino y 9 veces como Pava Andina.
- **Carpintero Candela:** Se identificó correctamente 197 veces. Fue confundida 15 veces con Águila Pechinegra, 17 veces con Clarinero Escarlata, 25 veces con Cóndor Andino y 60 veces con Pava Andina.

- **Clarinero Escarlata:** Fue clasificada correctamente en 128 ocasiones. Las confusiones más notables incluyeron 6 veces como Águila Pechinegra, 45 veces como Carpintero Candela, 26 veces como Cóndor Andino y 82 veces como Pava Andina.
- **Cóndor Andino:** Se identificó correctamente 214 veces. No obstante, fue clasificado erróneamente 57 veces como Águila Pechinegra, 10 veces como Carpintero Candela, 8 veces como Clarinero Escarlata y 6 veces como Pava Andina.
- **Pava Andina:** Fue clasificada correctamente en 210 ocasiones. Sus confusiones incluyeron 8 veces como Águila Pechinegra, 34 veces como Carpintero Candela, 6 veces como Clarinero Escarlata y 22 veces como Cóndor Andino.

El análisis detallado de las métricas de forma individual confirmó que las especies Cóndor Andino y Pava Andina fueron las mejor clasificadas por el modelo en escala de grises, mostrando una mayor capacidad de detección. Sin embargo, la especie Clarinero Escarlata representó la clase con mayor confusión y menor desempeño general en este modelo, siendo frecuentemente mal clasificada, lo que subraya las limitaciones del modelo en escala de grises para diferenciar visualmente esta especie basándose únicamente en la información de intensidad de píxeles. Las especies donde más se presentaron confusiones son aquellas que se diferencian por sus rasgos de color característicos, manifestando que la presencia de los tres canales permite al modelo identificar un conjunto de rasgos más amplio y discriminatorio, llevando a un rendimiento superior en la tarea de clasificación.

El desempeño general de este modelo fue inferior, pero se lo consideró un resultado moderado porque tiene la capacidad de generalizar las especies cuando son fotografías completas a una distancia cercana.

Figura 36

Matriz de Confusión del Modelo Escala de Grises

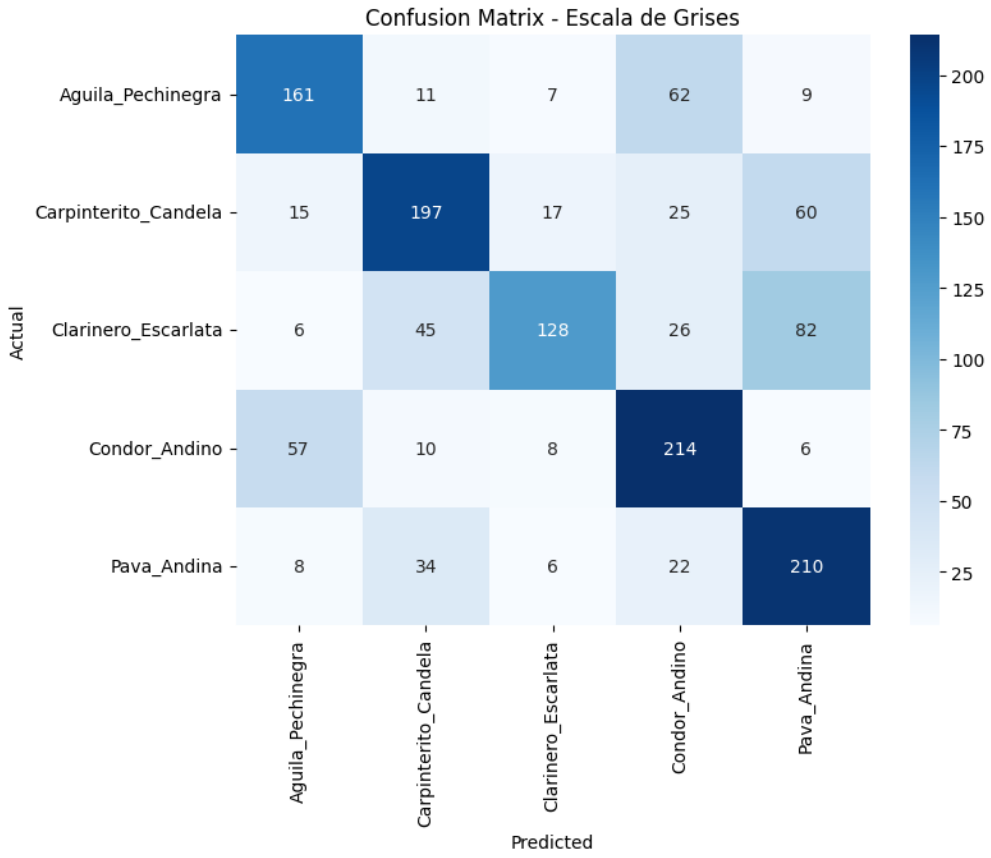


Figura 36: Matriz de Confusión del Modelo Escala de Grises

4.4 Capa de Visualización e Interacción del Modelo

El proceso de evaluación evidenció la diferencia significativa del modelo RGB para clasificar correctamente las aves incluso con imágenes fuera el conjunto de validación. Dado dicho resultado y con el fin de facilitar la comprensión y el uso de las capacidades predictivas del modelo a los usuarios finales, se implementó el prototipo de una capa de visualización utilizando el modelo con mejor rendimiento. Esta interfaz permite a los usuarios interactuar directamente con la red

neuronal, proporcionando una herramienta práctica para observar su funcionamiento en tiempo real y validar las clasificaciones obtenidas de manera intuitiva.

4.4.1 Despliegue

La etapa final del marco de trabajo contempló la integración del modelo en una interfaz desarrollada en el framework Django, configurando los componentes necesarios para su ejecución local. El sistema permite cargar imágenes nuevas, procesarlas en tiempo real y mostrar las predicciones correspondientes junto con datos propios de cada especie. Esta implementación se destinó exclusivamente a pruebas internas, enfocadas en validar la correcta interacción entre el modelo de clasificación y la capa de visualización, dejando las bases para una futura implementación dentro de un entorno de producción.

Es importante detallar ciertos aspectos claves que formaron parte del proceso de construcción del sistema final, los cuales se especifican a continuación.

4.4.1.1 Arquitectura y Diseño de la Interfaz. La capa de visualización se construyó siguiendo el patrón arquitectónico Model-View-Template (MVT), propio del framework Django. Este enfoque tiene la capacidad para separar de manera clara y estructurada la lógica de negocio, la gestión de datos y la presentación visual, lo cual facilita la implementación de cada componente de forma independiente.

Dentro de este patrón, el Model gestiona la interacción con la base de datos SQLite, donde se encuentra almacenada la información complementaria de cada especie de ave, organizada en la tabla “classifier_ave”. La View concentra la lógica de procesamiento, es decir, recibe la imagen enviada por el usuario, la prepara adecuadamente en base a los parámetros que la red neuronal

acepta, ejecuta la predicción utilizando el modelo RGB y consulta la base de datos para recuperar los datos asociados a la especie predicha. Finalmente, el Template genera dinámicamente el contenido HTML que se muestra en la interfaz web, integrando de manera comprensible el resultado de la clasificación y la información adicional obtenida.

Figura 37

Diagrama de la Arquitectura MVT del Sistema

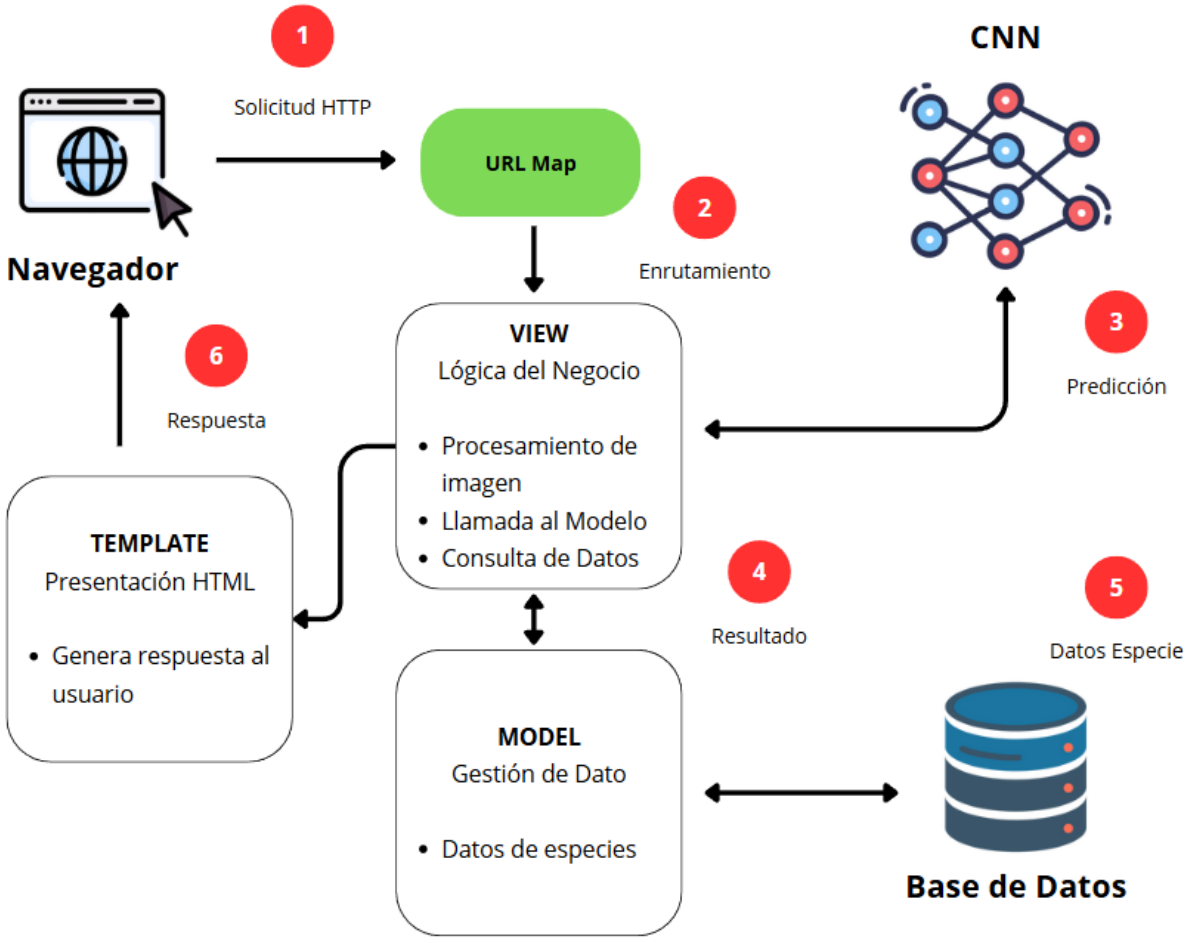


Figura 37: Diagrama de la Arquitectura MVT del Sistema

Una vez definida la arquitectura del sistema, se procedió a diseñar la capa de presentación mediante la elaboración de prototipos interactivos en Figma. La Figura 38 ilustra la estructura visual planificada para las principales vistas del sistema, incluyendo la página de inicio, el clasificador, el flujo de carga de imágenes y la pantalla de resultado con la información de la especie identificada. Estos mockups permitieron anticipar la disposición de los elementos de navegación, los formularios de interacción y la organización de los datos complementarios. Asimismo, el diseño preliminar tuvo una revisión y validación con biólogos expertos en la materia, definiendo criterios orientados a la claridad informativa, ajustes en la paleta de colores y la disposición de los componentes visuales de forma que la interfaz resultara intuitiva y accesible para todo tipo de usuarios.

Figura 38

Prototipo de Interfaz de Usuario Diseñado en Figma



Figura 38: Prototipo de Interfaz de Usuario Diseñado en Figma

4.4.1.2 Integración de Datos y Visualización Final. Luego de haber definido la arquitectura y diseño se procedió al desarrollo del sistema con todas las funcionalidades previamente establecidas. El primer punto clave fue cargar el modelo exportado en formato .h5 dentro del framework para habilitar la funcionalidad de clasificación automática de imágenes. La carga de este archivo se gestionó al iniciar la aplicación mediante la función "load_model", que se encarga de crear una instancia del clasificador lista para su utilización en tiempo de ejecución. De esta forma, la vista encargada del procesamiento puede invocar el modelo cada vez que un usuario envía una imagen, generando la predicción de manera inmediata sin requerir reentrenamiento.

Figura 39

Carga del modelo CNN entrenado en Django

```python
from tensorflow.keras.models import load_model

# Cargar el modelo previamente entrenado
MODEL = None
try:
    model_path = os.path.join(settings.BASE_DIR, 'models', 'modelo_aves_rgb.h5')
    MODEL = load_model(model_path)
except Exception as e:
    MODEL = None
```

Figura 39: Carga del modelo CNN entrenado en Django

Para complementar el proceso y buscando la forma de aportar valor a cada ave clasificada, se integró el almacenamiento y gestión de los datos descriptivos de cada especie mediante el modelo de datos definido en Django. Este modelo, denominado "Ave", permitió registrar de forma estructurada información adicional, como el nombre científico, estado de conservación, hábitat y mapa de distribución. Los registros se organizaron en la tabla "classifier_ave" y a través del panel de administración proporcionado por Django, fue posible realizar la carga inicial de datos y

administrarlos de manera visual, facilitando su consulta y modificación sin requerir intervenciones directas sobre el código o la base de datos. Esta implementación hizo posible que, tras la predicción de una imagen, el sistema recupere automáticamente la información complementaria correspondiente y la integre en la presentación final al usuario.

Figura 40

Panel de administración de Django con las especies registradas

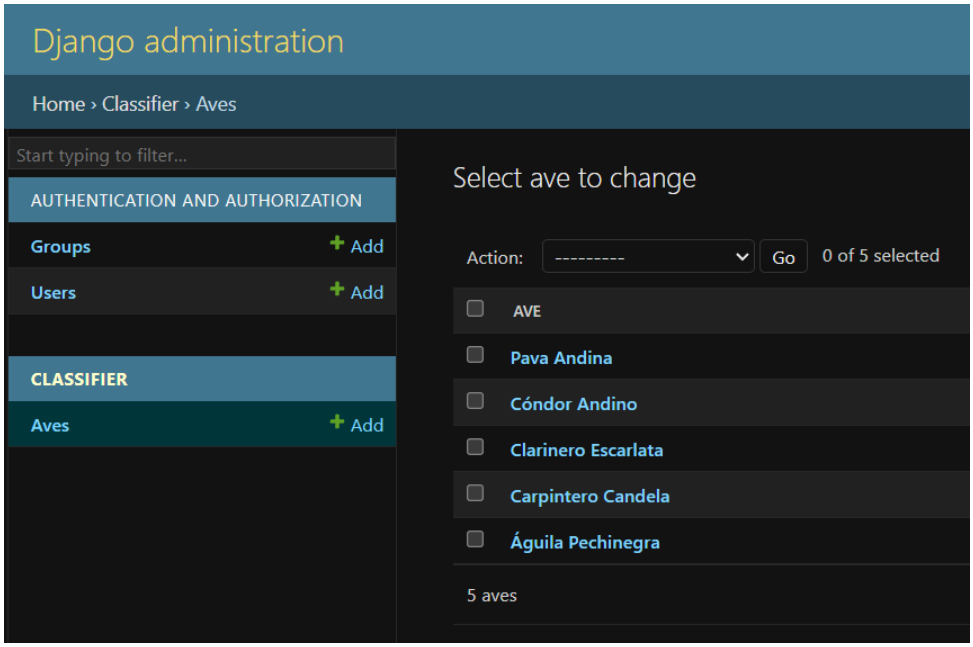


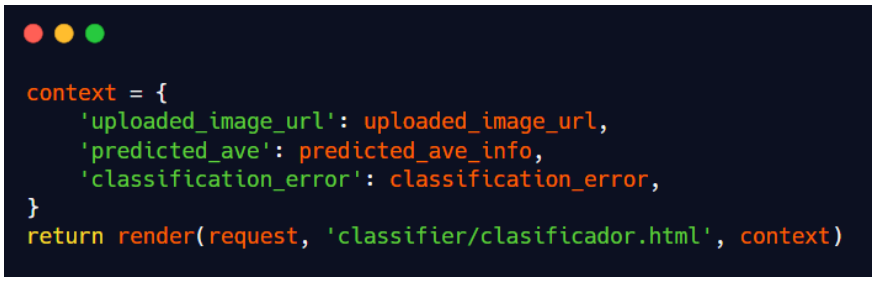
Figura 40: Panel de administración de Django con las especies registradas

El View es el componente central encargado de coordinar el flujo de clasificación, el cual permitió integrar la lógica de procesamiento de cada solicitud. Esta vista tuvo la responsabilidad de recibir las imágenes enviadas por el usuario, ejecutar las transformaciones necesarias para adecuarlas a las dimensiones y el formato esperado por el modelo y generar la probabilidad de pertenencia a cada clase. Una vez obtenida la especie identificada, toda la información resultante se organizó en un contexto estructurado que fue enviado al sistema de plantillas, permitiendo

construir la respuesta HTML que integra de forma unificada la imagen analizada, la predicción generada y los datos descriptivos de la especie.

Figura 41

Generación del contexto y renderizado de la respuesta



```
context = {
    'uploaded_image_url': uploaded_image_url,
    'predicted_ave': predicted_ave_info,
    'classification_error': classification_error,
}
return render(request, 'classifier/clasificador.html', context)
```

Figura 41: Generación del contexto y renderizado de la respuesta

Finalmente, se procedió a implementar de manera completa la capa de presentación del sistema, incorporando los estilos con CSS, la navegación y los elementos visuales definidos en el diseño aprobado. En primer lugar, se configuraron las plantillas responsables de renderizar las vistas principales, la cual incluyen la página de inicio, clasificador y el módulo informativo sobre la región andina. La interfaz fue presentada bajo el nombre KUNTUR ML, una propuesta personal que hace referencia al Cóndor Andino como símbolo de la biodiversidad de la región. Se adoptó una línea gráfica y tipografía sencilla, con el propósito de mantener la claridad visual y destacar los elementos más relevantes de la experiencia.

Figura 42

Pantalla de Inicio del Sistema KUNTUR ML



Figura 42: Pantalla de Inicio del Sistema KUNTUR ML

Para empezar el proceso de clasificación, el usuario se dirige a la pestaña de “Clasificador”, donde podrá subir la fotografía del ave que desee identificar. Tomando en cuenta que solo se puede proceder con una de las cinco especies con las cuales el modelo hizo su entrenamiento. Una vez seleccionada la imagen, el sistema procesa la solicitud y muestra la vista de resultados, en la que se exponen de manera estructurada los datos obtenidos. El resultado de la pestaña de clasificación incluye la imagen analizada, el nombre común y científico de la especie, el mapa de distribución geográfica y atributos adicionales como el estado de conservación y el hábitat característico.

Figura 43

Pantalla del Calsificador del Sistema KUNTUR ML

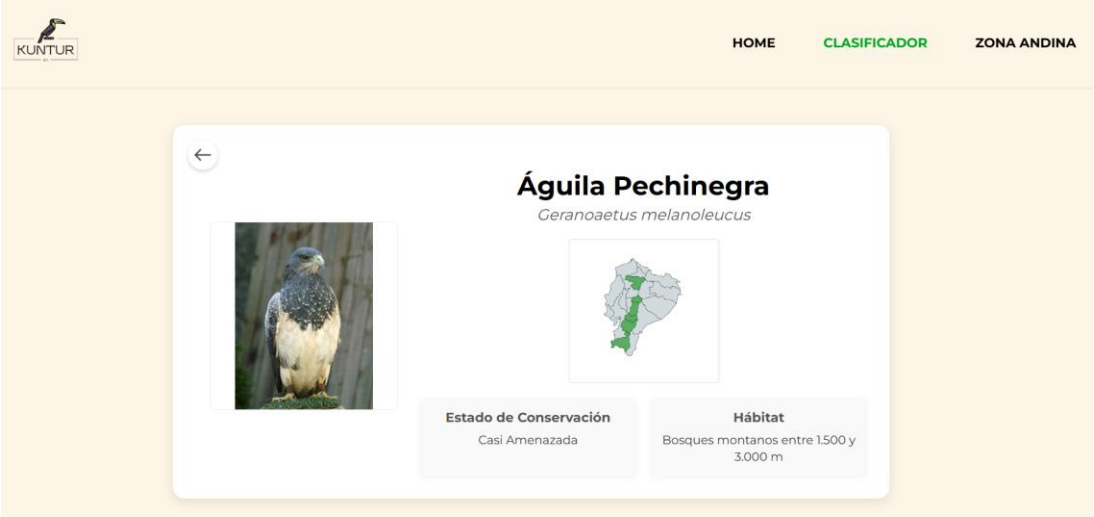


Figura 43: Pantalla del Calsificador del Sistema KUNTUR ML

La pestaña final del sistema es informativa, mostrando datos relevantes sobre la zona andina del Ecuador. Esta sección fue diseñada como complemento descriptivo que reúne información útil que fomenten el conocimiento sobre la diversidad de ecosistemas presentes en esta región. Esta característica sienta las bases para una futura implementación en entornos orientados a la divulgación científica y educación sobre la biodiversidad local.

Figura 44

Pantalla Informativa del Sistema KUNTUR ML

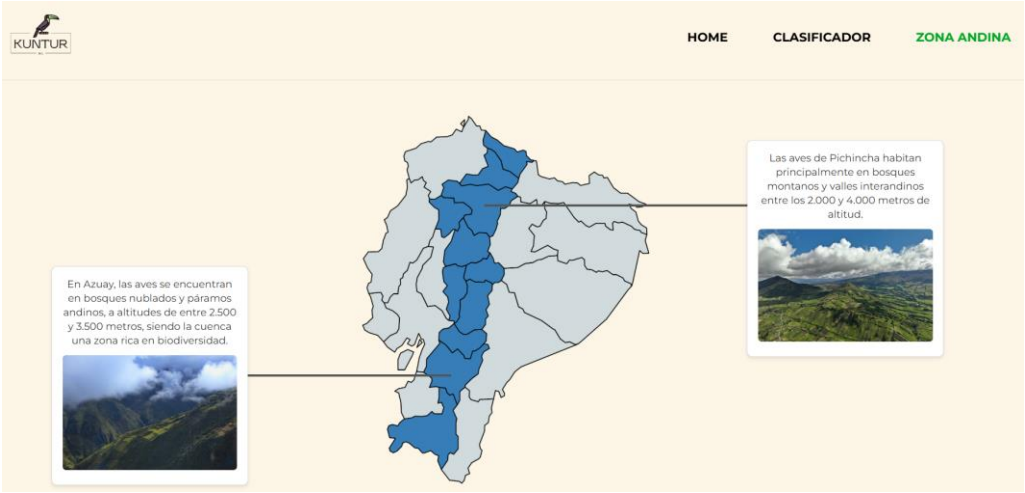


Figura 44: Pantalla Informativa del Sistema KUNTUR ML

CAPÍTULO V – CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

- El desarrollo del sistema de clasificación de aves de la zona Andina del Ecuador fue un completo éxito. Como resultado de este proceso, el modelo entrenado con imágenes RGB demostró ser el más efectivo, alcanzando una exactitud del 83.6%, superando significativamente al modelo en escala de grises, que obtuvo un 64%. Esta diferencia recalca que el uso de los tres canales de es fundamental para que el modelo aprenda características únicas específicas de cada ave y pueda generalizar mejor nuevo contenido.

- La adopción de la metodología CRISP-DM fue un factor determinante para la correcta ejecución del proyecto, al proporcionar un marco estructurado que guió cada etapa del desarrollo. Su capacidad iterativa entre fases permitió el ajuste continuo del modelo, la toma de decisiones fundamentadas y el establecimiento de criterios de validación claros, en busca de un resultado óptimo capaz de cumplir los objetivos propuestos.

- La calidad del conjunto de datos fue el pilar fundamental del proyecto y responsable directo del éxito del mismo. Concluyendo que la acumulación indiscriminada de fotografías no garantiza la obtención de un modelo robusto y eficiente. El proceso de selección manual de imágenes fue el responsable de otorgar valor real al entrenamiento, asegurando que las características visuales relevantes fueran claramente identificables y garantizando que el modelo pudiera aprender patrones consistentes que incrementaran su precisión y capacidad de generalización ante nuevos casos.

- La aplicación de técnicas de Data Augmentation fue un elemento esencial para optimizar la calidad y diversidad del conjunto de datos utilizado en el entrenamiento del modelo. La generación de variaciones artificiales permitió mitigar la limitación de un volumen reducido de muestras y reduciendo el riesgo de sobreajuste. Esto permitió la construcción de un conjunto de entrenamiento más representativo, que incluyera variaciones en orientación, iluminación y escalado.

- No existe un modelo único ni una arquitectura predeterminada que garantice el éxito en el entrenamiento de una red neuronal convolucional. El desarrollo del sistema permitió constatar que la búsqueda de un desempeño estable y preciso requiere un proceso iterativo de experimentación, donde es necesario probar distintas configuraciones de capas densas, técnicas de regularización, funciones de activación, número de épocas y tamaños de lote. Hasta encontrar una combinación de parámetros que mejor se adapte a las características del conjunto de datos disponible.

- El análisis de la matriz de confusión determinó que las especies Clarinero Escarlata y Carpinterito Candela demostraron ser las mejor clasificadas, exhibiendo una alta fiabilidad en sus predicciones, por otro lado, el Cóndor Andino se posicionó como la especie con más problemas, siendo frecuentemente confundida con el Águila Pechinegra.

- Django resultó una herramienta acertada para integrar el modelo de aprendizaje profundo en una aplicación web funcional y accesible. Su arquitectura basada en el patrón Model-View-Template permitió estructurar de manera ordenada la lógica de negocio, la gestión

de datos y la presentación visual, facilitando el desarrollo y la validación del prototipo. La capacidad del framework para escalar hacia entornos de producción más exigentes, sienta las bases para que el sistema pueda evolucionar en un futuro dentro de entornos de interés en la conservación y divulgación de conocimiento científico.

5.2 Recomendaciones

- Es recomendable evaluar con anticipación la capacidad computacional disponible antes de implementar técnicas de Deep Learning, debido a que muchas de sus librerías, como TensorFlow y Keras, requieren procesamiento acelerado mediante GPU y demandan recursos que generalmente superan las prestaciones de los equipos de uso convencional. En caso de no disponer de un hardware adecuado, resulta conveniente optar por entornos gestionados en la nube, como Google Colab, que ofrecen infraestructura optimizada para la ejecución de modelos de inteligencia artificial.

- Se recomienda siempre en proyectos que utilicen imágenes como insumo principal, priorizar la construcción de un conjunto de datos claro donde se resalte las características visuales de interés. Imágenes con exceso de ruido, sobreexposición, poca iluminación o elementos distractores, como vegetación o fondos confusos, pueden perjudicar la capacidad del modelo para aprender patrones relevantes y disminuir su desempeño. Por lo que resulta indispensable la selección manual de contenido del contenido de calidad sin importar el número de datos.

- Si el proyecto contempla su puesta en producción y la ampliación del número de especies disponibles, es recomendable que el modelo entrenado sea alojado en un entorno seguro y con acceso controlado, como un servicio de almacenamiento en la nube. La llamada al modelo debe ser gestionada a través de una API REST que reciba la imagen y devuelva la predicción, desacoplando el proceso de clasificación del servidor principal de la aplicación y facilitando su escalabilidad. La base de datos utilizada para almacenar la información complementaria de las especies debe migrarse a un sistema de gestión más robusto, como PostgreSQL, que permita manejar un mayor volumen de registros y ofrecer mecanismos de respaldo y recuperación ante fallos. Finalmente, la capa de visualización podría mantenerse como base de la interfaz web, pero requerirá ajustes en su configuración para ejecutarse de manera estable en un entorno de producción.

BIBLIOGRAFÍA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (pp. 265–283).

Chollet, F. (2018). Deep Learning with Python (2nd ed.). Manning Publications.

Cornell Lab of Ornithology. (s.f.). Merlin Bird ID. Recuperado de https://support.ebird.org/en/support/solutions/articles/48000966224-merlin-photo-id

Django Software Foundation. (s.f.). The web framework for perfectionists with deadlines. Recuperado de https://www.djangoproject.com/

Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Figma. (s.f.). Figma: The collaborative interface design tool. Recuperado de https://www.figma.com

Freile, J. F., & Cuervo, A. M. (2019). Avifauna of the Ecuadorian Andes: Diversity and conservation challenges. Ornitología Colombiana, 18, 17–38.

Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Hoyos, D., et al. (2021). Automated bird vocalization detection using machine learning. Journal of Acoustics and Environmental Studies, 30(3), 145–156. https://doi.org/10.1016/j.acoust.2021.03.015

IBM. (2020). IBM SPSS Modeler CRISP-DM Guide. IBM Corporation. Recuperado de https://www.ibm.com/docs/es/SS3RA7_18.4.0/pdf/ModelerCRISPDM.pdf

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

O'Gorman, L., & Kasturi, R. (2019). Document Image Analysis. IEEE Computer Society Press.

Quimis Barzola, A., & Ramírez Reyes, J. (2019). Sistema de clasificación, monitoreo y reporte de especies de aves utilizando redes neuronales convolucionales. Revista de Investigación en Ciencias Ambientales y Tecnologías, 7(6), 1464–1484. Recuperado de http://dominiodelasciencias.com/ojs/index.php/es/index

Ridgely, R. S., & Greenfield, P. J. (2001). The birds of Ecuador: Status, distribution, and taxonomy (Vol. 1). Cornell University Press.

Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

ANEXOS

- **Anexo 1.** Enlace al repositorio oficial del script inat_downloader.

  Disponible en: https://github.com/cypamigon/inat_downloader

- **Anexo 2.** Repositorio oficial del Sistema de Clasificación.

  Disponible en: https://github.com/isaakcueva/ClasificacionAvesCNN

- **Anexo 3.** Conjunto de datos – Dataset Aumentado RGB.

  Disponible en: https://www.kaggle.com/datasets/isaakcueva/dataset-aves-da-rgb

- **Anexo 4.** Conjunto de datos – Dataset en Escala de Grises.

  Disponible en: https://www.kaggle.com/datasets/isaakcueva/dataset-aves-da-escala-de-grises

- **Anexo 5.** Modelos CNN entrenados para clasificación de aves.

  Disponible en: https://www.kaggle.com/datasets/isaakcueva/modelos-cnn-clasificador-aves