UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Implementación de Redes Neuronales Convolucionales para la Identificación del Sexo de la Tilapia de Nilo *Oreochromis niloticus* a través de un Prototipo de Aplicación Móvil

Tesis

para obtener el título de:

Ingeniero en Computación

Presenta:

Guerrero Hernández Samuel Antonio

Director:

M.T.C.A. Moisés Emmanuel Ramírez Guzmán

Codirector:

Dr. Alfredo Gallardo Collí

Huajuapan de León, Oaxaca Julio de 2024

Dedicado a mis papás, Samuel y Soledad,
por todo el amor que me han dado.

Dedicado a mis hermanas, Mayra y Jessica
y al pequeño Sebastián.

Dedicado especial y con cariño para Alonso Eduardo.

Agradecimientos

A mi padre, Samuel Mario Guerrero Vázquez, y a mi madre, Juana Soledad Hernández García, por darme la oportunidad de estudiar esta hermosa carrera. Gracias a su sacrificio y esfuerzo, he logrado culminar esta gran travesía. Agradezco su amor y apoyo constante, tanto en los buenos como en los momentos difíciles. Les estaré eternamente agradecido por brindarme la mejor formación desde niño.

A mi director de tesis, M.T.C.A. Moisés Emmanuel Ramírez Guzmán, por su invaluable apoyo a lo largo de este proceso. Gracias por motivarme desde el inicio, por su tiempo, paciencia y vasto conocimiento; por confiar en mis capacidades y hacer posible este maravilloso trabajo. También le agradezco la oportunidad de haberme abierto las puertas al mundo laboral. Nada de esto sería posible sin su ayuda. Estaré infinitamente agradecido por todo lo que me ha brindado. Reciba toda mi admiración y respeto.

A mi codirector, Dr. Alfredo Gallardo Collí, por su generosa contribución de material, pruebas y conocimientos especializados. Agradezco profundamente su disposición para colaborar en la mejora de este trabajo, a pesar de la distancia. Al Dr. Eduardo Sánchez Soto, por su valiosa contribución de conocimientos y opiniones que han elevado la calidad de este trabajo. Estoy agradecido por los conocimientos compartidos durante las clases en la universidad y por los buenos momentos y risas que compartimos.

A mis hermanas, Mayra y Jessica, por haber concluido exitosamente nuestras carreras. Espero que nuestra experiencia sirva de inspiración para las futuras generaciones. A mi familia, especialmente a mis abuelas, por su apoyo incondicional a lo largo de todo este tiempo, así como al resto de mis familiares. A mi novia, Nayelli, por estar siempre a mi lado y por motivarme a continuar con este trabajo.

Este trabajo está dedicado con todo mi amor y cariño en memoria de Alonso Eduardo López Torres.

Índice general

Índice de figuras

Índice de tablas

Resumen

La clasificación de tilapias por sexo es un problema significativo dentro de la acuicultura, ya que el crecimiento y la productividad de las tilapias pueden variar según su sexo. En este trabajo, se propone una solución para este problema mediante el uso de Redes Neuronales Convolucionales (CNN) implementadas en dispositivos móviles.

La metodología presentada consta de tres etapas fundamentales: construcción de la base de datos, entrenamiento y selección del modelo, e implementación en dispositivos móviles. En la primera etapa, se construye una base de datos de imágenes de las papilas genitales de las tilapias, obtenidas gracias al personal de la Universidad del Mar (UMAR). Estas imágenes se preprocesan y dividen en conjuntos de entrenamiento y prueba.

En la segunda etapa, se exploran diferentes arquitecturas de CNN y se ajustan los hiperparámetros para maximizar el rendimiento del modelo. Se implementan técnicas de regularización y validación cruzada para garantizar que el modelo sea robusto y validar su capacidad de generalización. Finalmente y posterior a un análisis, el modelo con mayor rendimiento de clasificación es exportado para su uso en la última etapa.

En la etapa final, se carga el modelo en dispositivos móviles utilizando *React Native* y *TensorFlow.js*. Después de desarrollar y probar internamente el prototipo, se publica en *Google Play* para que sea accesible a cualquier persona con un dispositivo

Android. Además, se reportan pruebas de campo en cultivos de tilapias del Nilo para evaluar la precisión y capacidad del prototipo en condiciones reales de cultivo.

Este trabajo tiene el propósito de ser una aportación significativa al área de acuicultura, ya que proporciona una herramienta tecnológica eficaz y accesible para la clasificación de tilapias por sexo. Se espera que esta investigación contribuya a mejorar la productividad y eficiencia de los cultivos de tilapias en comparación de la clasificación tradicional que implica la presencia de expertos del área o uso de elementos como el azul de metileno.

Capítulo 1

Introducción

La acuicultura es el cultivo de organismos acuáticos, incluidos peces, moluscos, crustáceos e incluso plantas acuáticas. Esta disciplina se practica desde tiempos muy antiguos, pero apenas hace unas cuantas décadas comenzó a desempeñar un papel fundamental en la producción mundial de alimentos acuáticos, ya que más de la mitad de los productos destinados al consumo humano provienen de esta actividad. Con la intención de maximizar la producción, la acuicultura en nuestros días es una actividad multidisciplinaria que utiliza herramientas relacionadas con la biología, la ingeniería y la ecología. [FAO, 2023, Sun et al., 2020].

Una de las especies de peces más cultivadas es la tilapia del Nilo (*Oreochromis niloticus*), perteneciente a la familia *Cichlidae* [El-Sayed, 2006]. Esta especie es originaria de África y su cultivo se ha expandido a gran parte del mundo, con los principales productores ubicados en China, Indonesia, Egipto, Brasil y Tailandia. La tilapia se ha posicionado como la segunda especie más importante en términos de volumen de producción, solo superada por la carpa. Su capacidad de adaptación a una amplia variedad de ambientes, tanto naturales como en cautiverio, la hace notable por su tolerancia a diversas condiciones físicas, químicas y biológicas.

Esta diversidad de ambientes indica su capacidad de adaptación a un amplio espectro de condiciones físicas, incluyendo temperatura, fotoperiodo, profundidad, velocidad de la corriente y turbidez. Asimismo, demuestra su tolerancia a condiciones químicas variadas como salinidad, pH, oxígeno disuelto y concentraciones de minerales. En términos biológicos, la tilapia exhibe adaptabilidad a la competencia y disponibilidad de alimento, lo que permite su cultivo en jaulas flotantes instaladas en ríos o represas, así como en estanques de diferentes tipos, ya sean de tierra, concreto o geomembranas [Camacho Escobar, 2023].

Con el tiempo, la tilapia se ha convertido en la segunda especie más importante en términos de volumen de producción, solo superada por la carpa. Su capacidad reproductiva, rápido crecimiento, alta tolerancia a una gran variedad de ambientes y su alta resistencia al estrés y enfermedades permiten reducir los costos de crianza y producción de tilapias, generando precios más accesibles para los consumidores [Bhujel, 2014]. Según las estadísticas de la Food and Agriculture Organization (FAO), organización de la ONU, la producción mundial de tilapias pasó de 380,000 toneladas en 1990 a casi 6 millones de toneladas en 2018, con un aumento anual del 10.4 % [Martínez-Cordero et al., 2021].

En cautiverio, la tilapia se reproduce fácilmente sin necesidad de inyección de hormonas; este hecho se considera como su principal ventaja sobre otras especies [Bhujel, 2014]. La alta tasa de reproducción de las tilapias no siempre es una ventaja en todos los sentidos ya que puede conllevar a algunos inconvenientes; en condiciones naturales, la tilapia del Nilo madura entre los 150 y 200 gramos, mientras que en cautiverio la maduración puede ocurrir en tamaños tan pequeños como entre 30 y 50 gramos, acelerando su etapa reproductiva. El exceso de reproducción puede llevar a una sobrepoblación lo que provoca la competencia por los recursos como lo son el alimento, el oxígeno y el espacio [Perschbacher and Stickney, 2017]. Para abordar este problema es necesario clasificar a los especímenes reproductores por sexo para mantener una densidad poblacional más efectiva, especialmente en sistemas de cultivo con limitaciones de recursos.

La clasificación y su posterior separación es un método para crear poblaciones monosexuales (cultivos de un solo sexo) a una edad temprana con la finalidad de eliminar la reproducción incontrolada y sobre todo permitir sólo la producción de peces de tamaño comercializable [Alcántar-Vázquez et al., 2015]. Este proceso también puede ser empleado para obtener reproductores; esto es, elegir cuidadosamente a los individuos destinados a la producción de alevines que requiere de la separación de hembras y machos en una población mixta de peces [Meyer et al., 2015].

Existen varias maneras de generar poblaciones monosexuales de tilapias, una de ellas es la reversión sexual a través de hormonas exógenas que estimulan el cambio de sexo en los peces. La reversión sexual se ha convertido en el método más utilizado para tratar este problema y se ha extendido a casi todos los países donde se cultiva la tilapia [Bhujel, 2014]. Sin embargo, esta técnica ha generado una percepción negativa en los últimos años debido a los cambios genéticos que esta práctica puede ocasionar en los peces [Alcántar-Vázquez et al., 2015]. La decisión de cultivar poblaciones monosexuales no solo se toma para inhibir la reproducción en los estanques, sino también porque las tilapias presentan un crecimiento diferencial entre machos y hembras. En las hembras de tilapia, la precoz y constante madurez gonadal (producción de huevos) y la incubación maternobucal de los alevines involucran un alto gasto energético, lo que implica una proporción menor de energía dirigida al crecimiento en comparación con el gasto energético realizado en machos [Camacho Escobar, 2023].

La clasificación manual es otra forma de separar a estas poblaciones de tilapias, por medio de un análisis en el dimorfismo sexual que refiere a las diferencias visibles entre machos y hembras en términos de su fisonomía externa. En el caso de las tilapias, se presentan diferencias visibles en el tamaño, la forma o el color de ciertas partes del cuerpo, la forma de la cabeza o las características de sus papilas genitales [Meyer et al., 2015]. Este tipo de clasificación implica un análisis exhaustivo en la anatomía de los peces por parte de un especialista calificado, pero que puede derivar también en errores humanos por diversos factores como el cansancio o la experiencia del experto, además de que requiere de gran cantidad de tiempo [Bhujel, 2014].

En los últimos años, la Inteligencia Artificial (IA, por sus siglas en inglés) ha surgido como una herramienta prometedora con muchas aplicaciones en el sector acuícola, ya que puede facilitar mediante el uso de herramientas computacionales el desarrollo de algunas actividades generalmente reservadas a los humanos. La idea principal de la Inteligencia Artificial es la de emular computacionalmente la toma de decisiones que realizaría un ser humano. De manera ideal, se intenta emular el proceso del pensamiento humano, de tal manera que las máquinas tomen decisiones como los seres humanos inteligentes [Russell and Norvig, 2010].

Dentro del área de la IA, el Aprendizaje Automático (ML, por sus siglas en inglés) ha tenido un auge muy significativo en la última década. Es un campo científico que busca dotar a las máquinas de la capacidad de aprender y ha permitido obtener la solución a muchos problemas de clasificación en diversos ámbitos. La aplicación del ML en el ámbito de la acuicultura ha sido destacable, ya que los avances en las tecnologías de manejo de datos se están adecuando a resolver diversas dificultades que ha manifestado la acuicultura. Un ejemplo de estas aportaciones es el uso de herramientas de ML para solucionar problemas como la predicción de parámetros de calidad del agua para tomar mejores decisiones en los cultivos y en el diagnóstico de enfermedades de los peces para determinar tratamientos eficaces, entre otras problemáticas [Vásquez-Quispesivana et al., 2022].

Una de las herramientas del ML que ha tenido mayor aplicación en problemas de visión artificial son las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés). Las CNN han tenido una aplicación significativa en problemas de visión artificial, como clasificación [Krizhevsky et al., 2012] o segmentación de imágenes [Gong and Kan, 2021]. La aplicación de estas técnicas ha permitido resolver de manera eficiente problemas de clasificación o segmentación de imágenes, solo por mencionar algunas aplicaciones. Considerando lo antes mencionado, las CNN ofrecen un enfoque prometedor para abordar el desafío de la identificación sexual de la tilapia del Nilo. Su capacidad para procesar imágenes de manera eficiente en la tarea de separación de las poblaciones de hembras y machos las convierte en una valiosa herramienta para la gestión de cultivos acuícolas. Además de todas las bondades antes

mencionadas, una de las ventajas más destacadas de esta tecnología es su potencial para ser implementada en dispositivos móviles, lo que facilita su despliegue y uso práctico en el campo. La implementación de CNN en dispositivos móviles abre nuevas posibilidades para los acuicultores y gestores de cultivos de tilapias, pues este enfoque agilizaría el proceso de separación de machos y hembras, lo que, a su vez, mejora la eficiencia en la producción y planificación de la reproducción de tilapias.

1.1. Planteamiento del problema

En el cultivo de tilapia, la facilidad de reproducción en cautiverio es un aspecto crucial que permite obtener grandes cantidades de crías. Sin embargo, cuando se crían peces que no han sido sometidos a una reversión sexual hormonal, tienden a destinar una parte significativa de su energía alimentaria al desarrollo de gónadas. Esto afecta no solo la producción y calidad, sino también la uniformidad del tamaño de los peces [Trejo-Quezada et al., 2021].

Para evitar tales problemas, es importante hacer una selección sexual del lote de organismos en cultivo separando hembras de machos; este proceso precisa de un 80 % a 90 % de efectividad. Existen diversos métodos para abordar este problema; los dos principales se mencionan a continuación. El primer método se basa en la clasificación manual por observación de la papila genital. Este procedimiento requiere una cantidad significativa de mano de obra y además precisa de la experiencia de un experto, lo que puede tomar más tiempo de lo esperado al examinar grandes cantidades de peces. Debido a todas estas circunstancias y a pesar de la experiencia del personal que lleva a cabo la tarea, aún existe la posibilidad de error, especialmente en peces más pequeños [Beveridge and McAndrew, 2012]. El segundo método es la reversión sexual que permite la generación de poblaciones monosexuales por medio de hormonas aplicadas en los alimentos a los alevines. Este método puede implicar costos adicionales y además es una técnica poco ética con la naturaleza de las tilapias [Alcántar-Vázquez et al., 2015].

Considerando el crecimiento diferencial entre machos y hembras, es beneficioso hacer cultivos de poblaciones masculinas. En las hembras de tilapia, la precoz y constante maduración gonadal y la incubación maternobucal de los alevines implican un alto gasto energético, lo que resulta en una proporción menor de energía dirigida al crecimiento en comparación con la invertida en los machos. Esta es otra razón para cultivar poblaciones monosexo, lo que permite una mayor homogeneidad en las tallas y un mejor control del cultivo.

Además del control de la población de tilapias, la selección de reproductores es esencial para obtener crías de alta calidad. La clasificación precisa del sexo en las etapas iniciales del desarrollo de las tilapias es fundamental para asegurar la elección adecuada de ejemplares para su reproducción.

El objetivo principal de este trabajo de investigación es generar un conjunto de datos con imágenes enfocadas en las papilas genitales de tilapias del Nilo y desarrollar un modelo de ML basado en CNN para la identificación sexual para la separación temprana de los ejemplares antes que estos inicien su proceso reproductivo, así como para la selección de reproductores.

Se busca implementar este enfoque en dispositivos móviles, lo que no solo mejoraría la eficiencia en la producción y la elección de reproductores de alta calidad, sino que también haría que esta tecnología esté al alcance de un público más amplio en el sector acuícola. Esto significa que un mayor número de personas, independientemente de su experiencia visual en el campo de la acuicultura, pueden beneficiarse de esta aplicación.

1.2. Justificación

La acuicultura desempeña un papel fundamental en la alimentación de una población mundial en constante crecimiento. Este sector primario ha evolucionado no-

tablemente a lo largo del tiempo, adaptándose a diversas herramientas y tecnologías para maximizar la producción de especies como la tilapia del Nilo, una de las más relevantes en la industria.

La tilapia del Nilo se ha destacado debido a su capacidad de adaptación a diferentes entornos de cultivo, su menor exigencia en términos de calidad de agua y alimentación, y su alta tasa de reproducción. Sin embargo, este último aspecto ha generado desafíos significativos en los sistemas de cultivo. La rápida reproducción puede resultar en sobrepoblación y falta de control, lo que afecta la calidad, uniformidad y eficiencia del cultivo. Es crucial identificar y separar machos de hembras en una población mixta para mantener un equilibrio adecuado en la densidad poblacional, especialmente en sistemas con recursos limitados.

En este contexto, el presente estudio tiene como objetivo desarrollar una tecnología basada en CNN, implementada en un prototipo de aplicación móvil, que permita la identificación sexual de la tilapia del Nilo. Esta herramienta facilitaría la separación de machos de hembras en cultivos acuícolas donde no se utilice reversión sexual, ya sea para la obtención de reproductores destinados a la cría o para el monitoreo poblacional en cultivos sometidos a reversión hormonal.

Además de beneficiar a los acuicultores a gran escala, este prototipo de aplicación móvil también tendría un impacto positivo en aquellos con recursos limitados. Les proporcionaría acceso a una tecnología previamente inaccesible, permitiéndoles optimizar sus procesos de identificación sexual y mejorar la eficiencia de sus cultivos. Esto podría traducirse en un aumento significativo en sus ingresos y una mayor sostenibilidad en sus operaciones.

1.3. Hipótesis

La incorporación de Redes Neuronales Convolucionales en un prototipo de aplicación móvil resultará en una mejora significativa de la precisión en la identificación sexual de tilapias. Esta mejora se espera al compararla con los métodos tradicionales de clasificación manual por expertos, especialmente al utilizar imágenes focalizadas en las papilas genitales de las tilapias.

1.4. Objetivo general

Desarrollar e implementar un prototipo de aplicación móvil basada en Redes Neuronales Convolucionales que permita de manera eficiente y precisa la clasificación sexual de tilapias a partir de imágenes enfocadas en sus papilas genitales.

1.5. Objetivos específicos

Para lograr el objetivo general es necesario llevar a cabo los siguientes objetivos particulares:

- Revisar el estado actual de la aplicación de la visión artificial en la industria de la acuicultura y los procesos reproductivos relacionados al cultivo de tilapias.
- Realizar una revisión del estado actual en el campo de las Redes Neuronales Convolucionales.
- Crear una base de datos de imágenes enfocadas en las papilas genitales de tilapias de *Oreochromis niloticus*.

- Selección de una arquitectura adecuada para el modelo de la Red Neuronal Convolucional.
- Identificar los parámetros que maximicen la precisión del modelo basado en Redes Neuronales Convolucionales.
- Desarrollar el prototipo de aplicación móvil usando *React Native*.
- Publicar el prototipo de aplicación móvil en la tienda de *Google Play*.

1.6. Metas

Las metas establecidas para la realización del proyecto son:

- Elaboración de un informe sobre el estado actual de las Redes Neuronales Convolucionales.
- Creación de una base de datos a partir de imágenes enfocadas en las papilas genitales de tilapias del Nilo.
- Diseño y desarrollo un modelo basado en CNN para clasificar sexualmente tilapias del Nilo.
- Desarrollo y publicación en la tienda de Google Play de un prototipo aplicación móvil cargando el modelo de CNN.
- Redacción del documento de tesis.

1.7. Limitaciones

Las limitaciones del proyecto de tesis son:

- El modelo de CNN será entrenado solamente con imágenes fijas de la base de datos elaborada.
- El sistema puede predecir a partir de machos con un peso de 100 gramos y en hembras a partir de 50 gramos.
- El sistema estará restringido a la clasificación de imágenes de papilas genitales de tilapias del Nilo y sus subespecies relacionadas.
- El prototipo de aplicación móvil realizada por medio de *React Native* podrá ser usada únicamente en sistemas operativos *Android*. La aplicación podrá hacer predicciones a partir de imágenes capturadas por la cámara del dispositivo móvil.

1.8. Trabajos relacionados

La industria de la acuicultura en los últimos años ha requerido de soluciones a diversas problemáticas relacionadas con la clasificación o conteo de especies en cultivos. Muchas de estas problemáticas se han logrado tratar recientemente con herramientas del área de la inteligencia artificial y más concretamente con técnicas de aprendizaje profundo. A continuación, se presentan algunos ejemplos de los trabajos recientes y más destacados relacionados con este problema.

La medición individual de rasgos en peces busca mejorar la producción acuícola y la calidad de la carne. Existen dos métodos para abordar este problema: el primero consiste en la medición manual de los peces, lo cual implica sacarlos del agua para medir sus rasgos individualmente. Este método puede ser bastante lento para cultivos grandes. Para lograr la medición de los rasgos, [Fernandes et al., 2020] presentaron un sistema de visión artificial. Este sistema utiliza redes de aprendizaje profundo para segmentar imágenes de *Oreochromis niloticus* en tres clases: fondo, aletas de pez y área del cuerpo. Luego, se extraen medidas morfométricas, como el área del cuerpo, longitud, altura y excentricidad. Estas medidas se utilizan en modelos lineales

para predecir el peso corporal y el peso de la canal de los peces, con altos niveles de precisión (R2 de 0.96 y 0.95, respectivamente). El sistema de visión artificial proporciona una solución rápida y precisa para medir los rasgos de interés sin estresar a los peces, lo que mejora la forma en que se realiza esta tarea.

El conteo de peces es una actividad fundamental para la estimación de biomasa en la acuicultura, además de que permite lograr una alimentación razonable y, en consecuencia, una mejor eficiencia en la reproducción. Sin embargo, los métodos tradicionales de conteo de peces requieren la detección individual de cada uno de ellos, lo que puede resultar complicado debido a la complejidad de las imágenes de los peces y su procesamiento. Para resolver este problema, [Zhang et al., 2020] presentan un sistema de visión artificial para la clasificación de peces que utiliza clasificación de densidad de imagen y regresión local, logrando un error medio absoluto de 0.2985, un error cuadrático medio de 0.6205 y un coeficiente de precisión del 96.07

La detección de peces es crucial para el monitoreo de poblaciones acuáticas, y su complejidad ha llevado a la aplicación de técnicas de visión artificial. [Almero et al., 2019] proponen un enfoque que integra una CNN en un algoritmo de visión artificial. Esto permitió analizar imágenes de peces desde diferentes ángulos y extraer características relevantes para la detección. Los resultados muestran un error cuadrático medio de 0.2315 y una precisión del 79.00 %.

En otro ámbito de la acuicultura, la determinación de la frescura de los ejemplares también ha sido objeto de investigación y estudio dentro de la Visión Artificial. La frescura de las distintas especies de peces consumidas por el humano es fundamental en la aportación nutricional al consumidor, caso contrario a cuando un ejemplar tiene varios días de haber sido capturado y su frescura es casi nula, lo que puede derivar en una intoxicación alimentaria. Este problema se abordó por [Navotas et al., 2018]; en dicho trabajo, se propuso una aplicación *Android* que identifica automáticamente las tres especies de pescado más consumidas en Filipinas: *Chanos chanos, Decapterus Punctatus y Oreochromis niloticus*, a través del procesamiento de imágenes y RNA utilizando los valores RGB de los ojos y branquias. Esto permite clasificar la frescura

en una escala que va desde el nivel 1 (rancio) hasta el nivel 5 (fresco). El sistema utilizó una red neuronal preentrenada de tipo *feed-forward*, y los resultados obtenidos fueron notables: precisión del 90 % para *Chanos chanos*, 93.33 % para *Decapterus Punctatus* y 100

La detección de enfermedades en los peces también se encuentra dentro de las áreas de aplicación de la visión artificial. Estas enfermedades pueden propagarse rápidamente en los cultivos a través del agua, lo que las convierte en un problema significativo en la acuicultura. Tradicionalmente, las enfermedades en los peces han sido diagnosticadas manualmente por acuicultores expertos a simple vista, pero en otras ocasiones se requieren análisis de laboratorio para determinar las causas o microorganismos que ocasionan dichas enfermedades; dicho proceso implica tiempos que en ocasiones son excesivos y también un aporte económico importante, lo cual afecta directamente a la infraestructura de los cultivos. Para abordar esta problemática, [Hasan et al., 2022] proponen un sistema utilizando CNN para detectar y clasificar el tipo de enfermedades entre los peces infectados. El estudio se enfocó en tres clases: peces sanos, peces con mancha roja y peces con mancha blanca. Se alcanzó una precisión de 94.44

Ahondando en el problema de detección de enfermedades, el siguiente trabajo se enfoca en un caso particular, "Hibay". Esta enfermedad es generada por *Streptococcus agalactiae*, una bacteria patógena, que tiene un impacto significativo en las poblaciones de peces, particularmente en las tilapias del Nilo (*Oreochromis niloticus*). Esta enfermedad es de alto riesgo ya que presenta una propagación infecciosa veloz, derivando en pérdidas masivas de peces en poblaciones, lo que también provoca pérdidas económicas; esta enfermedad afecta el sistema nervioso central de los peces, lo que provoca alteraciones en su comportamiento y en su fisonomía. Además, el consumo de peces infectados con esta enfermedad puede causar infecciones graves. Para poder abordar este problema,[Hernandez and Hernandez, 2019]) proponen un sistema basado en CNN usando el modelo Inception V3. Este sistema demuestra la capacidad de diferenciar de manera efectiva entre las tilapias infectadas con Hibay y aquellas que no presentan la enfermedad. Los resultados obtenidos de esta propuesta

son notablemente positivos, alcanzando una tasa de precisión del 100

En la cría de guppies (*Poecilia reticulata*), separar los sexos es crucial tanto para controlar las poblaciones como para garantizar la reproducción selectiva y mantener características genéticas deseables. La calidad de los guppies se determina por las características de forma y color presentes en sus cuerpos y aletas, las cuales difieren entre los peces hembra y macho. Para abordar la clasificación por sexo de los guppies, [Zion et al., 2008] propusieron un sistema que emplea algoritmos de procesamiento de imágenes para analizar las características de forma y color en imágenes de *Poecilia reticulata*. Las pruebas realizadas usando la forma de los peces lograron un 90 % de precisión, mientras que para el color se alcanzó un 96 % de precisión. Al combinar ambas características, la precisión llegó hasta un 98 % de efectividad.

Actualmente, no se ha implementado un sistema de clasificación de tilapias por sexo utilizando herramientas de ML. El único método documentado es a través de una exploración visual de la papila genital de los especímenes por un experto. Por otro lado, la forma más eficiente para obtener poblaciones monosexuales es aplicando reversión sexual hormonal; este proceso implica costos extras, además una gran parte de los consumidores tiene renuencia a consumir peces tratados usando este método [Alcántar-Vázquez et al., 2015]. Dados los puntos anteriores, se considera de gran importancia el tratamiento de este problema usando técnicas de ML.

1.9. Metodología

Para desarrollar una CNN destinada a la clasificación de tilapias por sexo en dispositivos móviles, se sigue una metodología específica que se detalla en la Figura 1.1. Esta metodología consta de tres etapas fundamentales, que se describen a continuación.

En la primera etapa, el objetivo es la construcción de una base de datos que

permita el entrenamiento de la CNN. Los pasos propuestos para la construcción de la base de datos se detallan a continuación:

a. **Adquisición de las imágenes:** El conjunto de imágenes enfocadas en las papilas genitales de ejemplares de tilapia del Nilo se obtuvo gracias al personal de la Universidad del Mar (UMAR) en diversas sesiones con ejemplares cultivados en dicha institución.

b. **Preprocesamiento de Imágenes:** En esta fase, las imágenes adquiridas se someten a un proceso de mejora y selección. Primero, se recortan para eliminar cualquier contenido innecesario, manteniendo únicamente las áreas enfocadas en las papilas genitales de las tilapias. Luego se descartan aquellas imágenes que tienen mala calidad y no son aptas para entrenar la red, garantizando que solo las imágenes de buena calidad se utilicen en el proceso de clasificación por género.

c. **División de datos:** En esta etapa, las imágenes procesadas se dividen en dos conjuntos distintos: uno de entrenamiento y otro de prueba. El conjunto de entrenamiento se utiliza para entrenar la CNN, permitiendo que aprenda a identificar las diferencias de género en las tilapias. El conjunto de prueba se reserva para evaluar la capacidad de generalización del modelo basado en CNN.

En la segunda etapa, el objetivo es entrenar y crear N modelos $M_1, M_2, \ldots, M_N$. De estos modelos, se selecciona el mejor utilizando métricas de rendimiento como criterio de evaluación. Esta etapa comprende los siguientes puntos.

a. **Selección de Arquitectura de CNN:** Durante esta fase, se exploran diferentes arquitecturas de CNN en el estado del arte con el objetivo de determinar cuál se adapta mejor al problema.

b. **Ajuste de hiperparámetros:** El objetivo de esta fase es encontrar la combinación óptima de parámetros mediante una búsqueda en malla de los valores que maximicen el rendimiento del modelo en el conjunto de datos de validación.

c. **Regularización:** En esta fase, se busca aumentar el tamaño del conjunto de datos obtenido en la etapa anterior aplicando transformaciones a las imágenes con el fin de mejorar el entrenamiento de las CNN.

d. **Validación cruzada:** Para evaluar y comparar el rendimiento de varios modelos, se implementa la validación cruzada estratificada. Esto proporciona una evaluación más robusta del rendimiento de los modelos. Al final de la validación cruzada, se compararán y analizarán los rendimientos de clasificación de un modelo con cada uno de los K folds. El modelo seleccionado será aquel que tenga el mejor rendimiento de clasificación en el k-ésimo fold, determinado por su exactitud (accuracy).

En la tercera etapa, el objetivo es implementar el modelo elegido en la fase anterior en dispositivos móviles. Esta etapa contiene los siguiente puntos.

a. **Exportación del modelo:** Una vez que el mejor modelo ha sido entrenado y seleccionado en la fase anterior, se procede a exportar este modelo utilizando TensorFlow. Esto implica la generación de dos archivos, uno en formato *h5* y otro en formato *JSON*. Posteriormente, estos archivos se transforman con TensorFlow.js para generar dos archivos adicionales en formato binario y *JSON*. Estos archivos permiten cargar el modelo entrenado en entornos basados en *JavaScript* y *TypeScript*.

b. **Carga del modelo en *React Native* y construcción de la aplicación móvil:** Una vez que se han obtenido los archivos resultantes del paso anterior, es posible cargar el modelo en dispositivos móviles utilizando *React Native*. *React Native* es un framework que se enfoca en el desarrollo de aplicaciones móviles utilizando *JavaScript* y *TypeScript*. La carga del modelo se logra a través de la API *tfjs-react-native*, que permite integrar modelos entrenados

con *TensorFlow.js* en aplicaciones *React Native*. Esto habilita la ejecución del modelo en dispositivos móviles para realizar inferencias en tiempo real. Este paso implica también la construcción de la interfaz gráfica del prototipo por medio de *CSS* y *TypeScript*. Después de implementar la aplicación móvil, se llevarán a cabo pruebas de campo para probar el prototipo en cultivos de tilapias del Nilo, y se analizará su rendimiento.

c. **Publicación en Google Play:** Una vez que el prototipo ha sido desarrollado y probado internamente, se procede a publicarla en la tienda de aplicaciones de Google Play. Esto permite que el prototipo sea accesible para cualquier persona con un dispositivo *Android*.

d. **Pruebas de campo en cultivos de Tilapias del Nilo:** Posterior a la publicación en *Google Play*, se llevarán a cabo pruebas de campo para probar el prototipo en cultivos de Tilapias del Nilo. Durante estas pruebas, se analizará el rendimiento del prototipo en un entorno real, evaluando su precisión y capacidad para clasificar las tilapias por sexo en condiciones reales de cultivo.

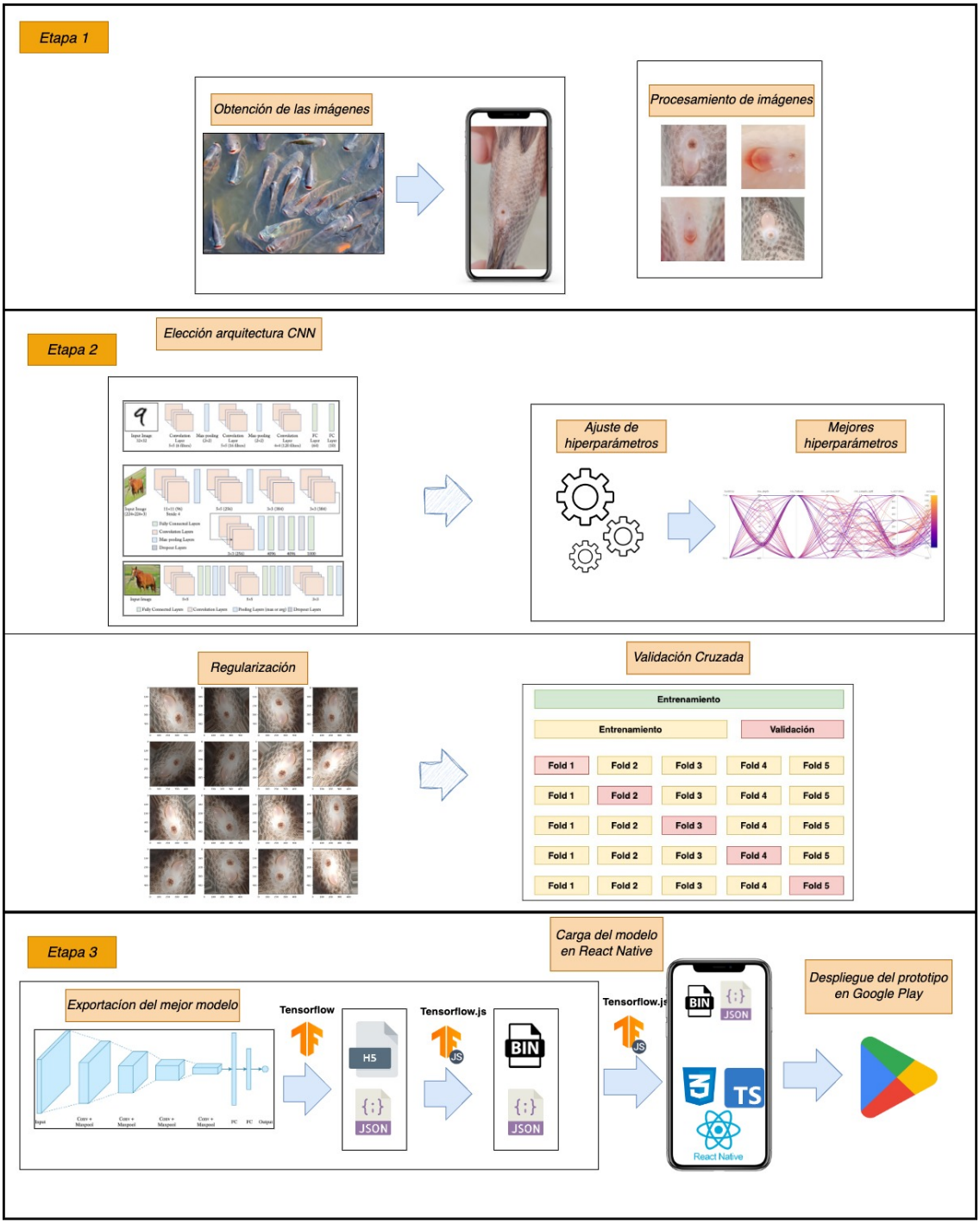


Figura 1.1: Metodología para entrenar e implementar una CNN en dispositivos móviles.

Capítulo 2

Marco Teórico

En el presente capítulo se abordan conceptos fundamentales para el desarrollo de esta tesis. En la Sección 2.1 se analiza la anatomía de la tilapia del Nilo y sus etapas de reproducción que influye directamente la maximización de la talla esperada de los ejemplares en un cultivo. En la Sección 2.2, se abordan los conceptos esenciales del aprendizaje automático, centrándose en el problema de clasificación binaria. En la Sección 2.3 se habla acerca de las CNN y los componentes clave de su arquitectura y funcionamiento. En la Sección 2.4 se analizan las técnicas más importantes que permiten ajustar los parámetros de los modelos durante su entrenamiento. La Sección 2.5 explora las métricas relevantes en el contexto de la clasificación binaria, proporcionando herramientas para evaluar y cuantificar el desempeño de los modelos desarrollados. Por último, en la Sección 2.6, se examinan las bibliotecas de desarrollo que desempeñarán un papel fundamental en la implementación de este proyecto.

2.1. Dimorfismo y ciclo reproductivo de la tilapia

La tilapia muestra un alto grado de dimorfismo sexual en comparación con la mayoría de las otras especies de peces utilizadas en la acuicultura. Los machos y las hembras pueden identificarse basándose en la morfología genital externa , en particular, la papila genital ubicada posterior al ano [Beveridge and McAndrew, 2012]. En los machos, esta papila se presenta alargada y puntiaguda, y en su extremo distal se observa el orificio de la uretra, utilizado para la eliminación de orina y la liberación de fluido seminal [Bhujel, 2014].

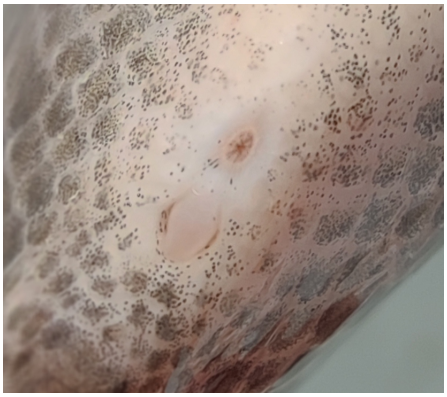


Figura 2.1: Papila genital de tilapia macho.

En el caso de la papila genital de la hembra, suele ser más redonda y menos puntiaguda en contraste con la del macho, además de presentar un tono rojizo [Bhujel, 2014]. Cerca de la base de la papila se observa también el orificio del oviducto de la hembra en el cual son expulsados los huevos durante el desove. En el caso de hembras vírgenes el oviducto suele ser menos visible en la zona de la papila genital lo cual dificulta la identificación de su sexo a temprana edad [Meyer et al., 2015].

Las diferencias debidas al dimorfismo sexual en las tilapias permiten su clasificación mediante un análisis visual de las papilas genitales. Este método requiere una evaluación visual de las papilas urinogenitales, identificando el número de aberturas para distinguir entre machos y hembras. Aunque esta técnica es relativamente sencilla, resulta extremadamente laboriosa y estresante para los peces. La precisión del

sexado manual oscila entre el 80 y el 90 %, y cuando son peces de menos de 10g la tendencia es clasificar como hembras a los machos, generando así mayor número de errores [Penman and McAndrew, 2000, El-Sayed, 2006].

Figura 2.2: Papila genital de tilapia hembra.

La tilapia se adapta perfectamente a vivir en agua salina y salobre la cual se refleja en su presencia en muchos cuerpos de agua presentes en el mundo. La tilapia se reproduce durante todo el año, ya sea de manera espontánea o en cautiverio, y no requiere de condiciones ni estimulaciones especiales; sin embargo, existen factores importantes para que se logre una mayor eficacia en la reproducción, uno de ellos son la temperatura ya que se recomienda un rango de entre 24°C y 32°C puesto que a temperaturas menores, el crecimiento suele ser más lento y el cultivo se vuelve poco rentable. La altura también es importante ya que se recomiendan niveles menores a 1200 *msnm* (metros sobre el nivel del mar), cabe notar que hay una relación directa entre la altura y la temperatura, ya que a mayor nivel del establecido los climas suelen ser más frescos y fríos, lo que impide un óptimo crecimiento de los alevines y tilapias jóvenes. También se recomienda un nivel de pH en un rango de 6.5 a 9.0 al igual de una buena transparencia en el agua, ya que las tilapias suelen comunicarse visualmente y es importante para el cortejo [Meyer et al., 2015].

La madurez sexual de la tilapia es alcanzada entre los 4 y 6 meses de vida, aunque existe cierta precocidad en tilapias de tan solo 3 meses dada la naturaleza de

la especie, lo cual deriva en problemas en las tallas finales de los peces. La tilapia es una especie que se reproduce con bastante facilidad y se caracteriza por tener una gran capacidad reproductiva dadas las bajas exigencias de su entorno reproductivo. La tilapia hembra suele mantener huevos de distintos estados de madurez en su tracto reproductivo y en el momento de cada cortejo sólo desova los huevos de mayor madurez. Por el lado del macho, éste suele construir el nido en el cual la hembra desova los huevos y comúnmente está ubicado al fondo de los cuerpos de agua. Al ser una especie agresiva, suelen defender el territorio aledaño al nido [Meyer et al., 2015].

Los machos son capaces de identificar correctamente a las hembras dispuestas a iniciar la etapa reproductiva y cotejarlas. Una vez formada una pareja suelen nadar constantemente cerca del nido en un acto de apareamiento que puede durar desde minutos hasta horas en el que el macho suele estar en la parte superior de la hembra. Acto seguido la hembra suele depositar los huevos en el nido y posteriormente el macho libera el esperma sobre los huevos para ser fecundados; el proceso suele ocurrir durante varias ocasiones hasta que no haya más huevos maduros en su tracto reproductivo. Finalmente, la hembra recolecta todos los huevos junto al esperma en su cavidad bucal para después alejarse del macho e iniciar la etapa de incubación que suele durar entre 5 a 15 días en el que influye directamente la temperatura del agua. Durante ese lapso del tiempo la tilapia hembra no suele ingerir alimentos por la presencia de los huevos en la cavidad bucal lo que deriva en un atraso en su crecimiento final al no haber ganancia de peso. Una vez eclosionados los huevos, los pequeños alevines abandonan la boca de la madre para empezar su vida independiente en cardúmenes en las partes superiores del estanque [Meyer et al., 2015].

2.2. Aprendizaje automático

El Aprendizaje Automático (ML por sus siglas en inglés del término *Machine Learning*) es una rama de la inteligencia artificial que tiene como objetivo el diseño de métodos de aprendizaje automáticos que otorguen a las entidades de cómputo la

capacidad de aprender por medio de la interacción con el mundo real mediante enfoques como la visión artificial. Un punto importante del ML es que las computadoras aprenden a partir de la abstracción de datos o características otorgadas de entes u objetos del mundo real sin una programación explícita, sin una definición explícita de reglas o lógica por parte de los humanos [Khan et al., 2018]. Estos métodos de aprendizaje están divididos en tres principales enfoques: supervisados, semi-supervisados y no supervisados. Y otros autores como [Sutton and Barto, 2018] contemplan además el aprendizaje por refuerzo como un cuarto enfoque.

Los métodos de aprendizaje supervisados son los más populares y los más usados por la industria en la actualidad debido a su rendimiento superior en comparación a los otros métodos mencionados. Este tipo de método se caracteriza por los datos utilizados en el entrenamiento que consta de una colección de información conformada por un par de conjuntos X : *característica*, Y : *clase* que determinan una relación entre algún elemento de X y un elemento de Y con una correspondencia unívoca representada por [Khan et al., 2018]:

$$f : X \to Y$$

Es decir, es una correspondencia en donde cada elemento del conjunto de X : *característica* corresponde a uno y solo un elemento del conjunto imagen Y : *clase*, por lo tanto los métodos de aprendizaje supervisados son entrenados a partir de elementos que no pueden pertenecer a más de una clase al mismo tiempo, como se muestra en la Figura 2.3.

En la imagen se aprecia la relación entre el conjunto de características y el conjunto de clases. En este contexto, las imágenes de perros y gatos representan dos clases distintas. Sin embargo, existen problemas que involucran más de dos clases, conocidos como problemas multiclase. En estos casos, un conjunto finito de elementos en los datos corresponde a un número también finito de clases en las etiquetas. Esto amplía la clasificación más allá del binomio perros-gatos, permitiendo conside-

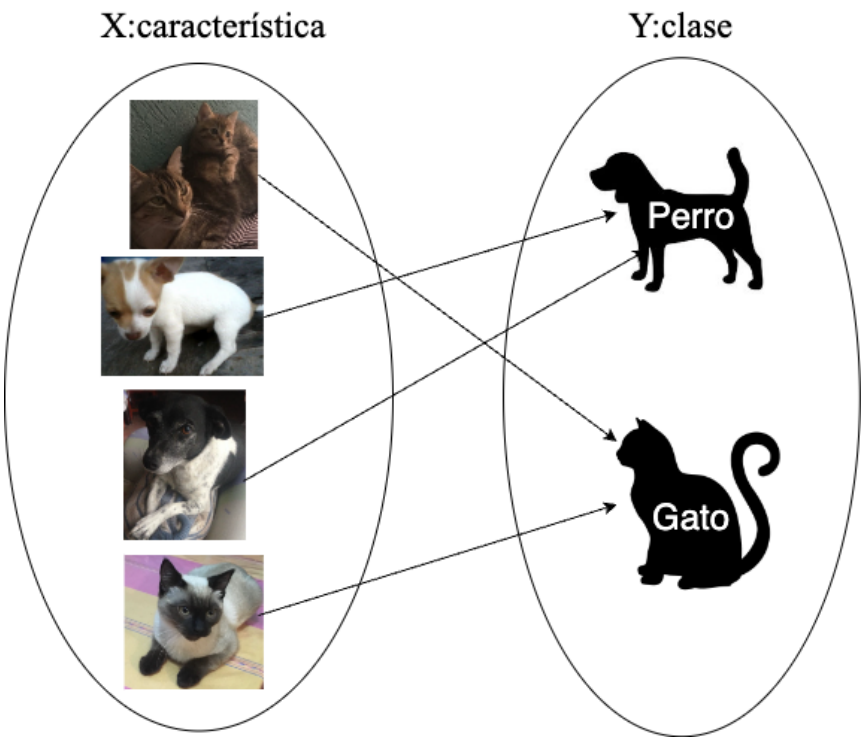


Figura 2.3: Relación unívoca (analogía entre perros y gatos).

rar una variedad más amplia de animales, como aves, anfibios, reptiles, mamíferos, entre otros.

Los métodos de aprendizaje supervisados están diseñados para aprender a clasificar durante la etapa de entrenamiento, al ajustar sus parámetros. Estos buscan aproximar una función $f(x)$ que pueda predecir los valores de y pasando por argumento un elemento de x, la cual puede ser resuelta por diversos métodos de aprendizaje supervisado como lo son los *árboles de decisiones*, *bosques aleatorios* (RDF), *regresión logística* (LR), *máquinas de vectores de soporte* (SVM), *redes neuronales* (NN), y *clasificadores Bayesianos*, entre otros [Khan et al., 2018].

2.3. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) son una de las categorías más populares y útiles en el mundo de las redes neuronales, es utilizado generalmente para tareas de visión artificial. Éstas surgieron tras el estudio del córtex visual del ojo motivado por el trabajo de [Hubel and Wiesel, 1959] por medio del *neocognitrón*. En este experimento se demostró que las neuronas en el cerebro están formadas por capas que aprenden a reconocer patrones visuales extrayendo primero las características locales y luego combinándolas para obtener representaciones de mayor nivel. El entrenamiento de la red se realizó mediante el aprendizaje por refuerzo. Años más tarde, en 1989 se realizó una mejora importante sobre el *neocognitrón* por medio del modelo *LeNet* propuesto por *LeCun* donde los parámetros del modelo se ajustaron gracias al algoritmo de *propagación hacia atrás* (del inglés, *backpropagation*) el cual se aplicó exitosamente para reconocer dígitos escritos a mano [LeCun et al., 1998]. Actualmente, las redes neuronales son usadas en múltiples tareas cotidianas y sectores de la industria por su gran eficiencia y utilidad, pues tiene aplicaciones en la medicina, biología, industria alimentaria, vigilancia y seguridad, entre otras [Li et al., 2021].

Una CNN aprende a asignar una imagen determinada a su categoría correspondiente mediante la convolución, pues en esta fase se extrae una serie de características abstractas que van desde lo más simple hasta lo más complejo, como bordes, texturas, formas, coloraciones, etc. Las capas convolucionales están intercaladas con capas de agrupación (del inglés *pooling*) para reducir significativamente la cantidad de variables que pueden aprenderse [Li et al., 2021]. Después de varias capas convolucionales y de agrupación, las características extraídas son enviadas a una capa de aplanamiento (*flatening*) para posteriormente ser enviadas a una red completamente conectada, implementada por un perceptrón multicapa (MLP, por sus siglas en inglés), que finalmente asigna una clase a la imagen o video de entrada. Las CNN han sido muy efectivas en una gran variedad de tareas de visión artificial, incluyendo la clasificación de imágenes, el reconocimiento de objetos, la segmentación de imágenes, generación de imágenes. Adicionalmente, hay aplicaciones documentadas

en problemas de reconocimiento de voz, procesamiento del lenguaje natural, entre otros.[Khan et al., 2018, Li et al., 2021].

Una CNN se compone de varios bloques en su arquitectura llamados capas de CNN. A continuación se muestran a detalle los componentes básicos y su función:

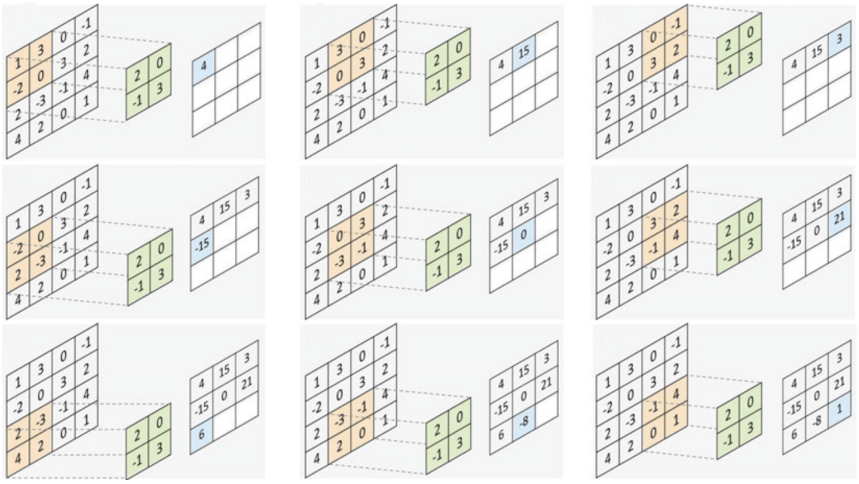
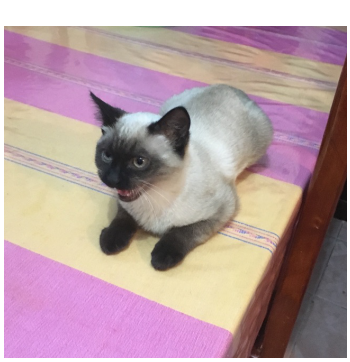


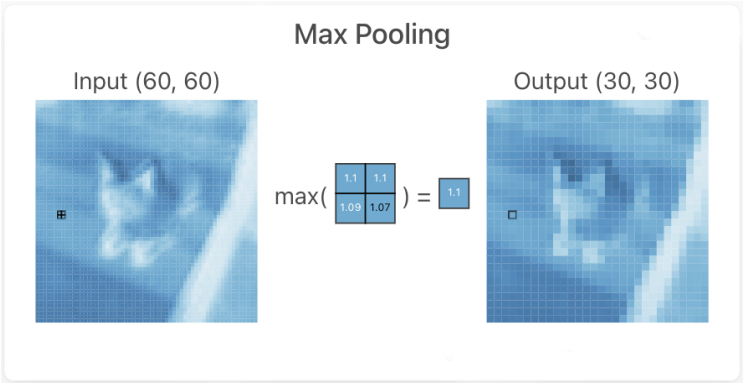
Figura 2.4: Operación de convolución aplicada a un mapa de entrada para generar un mapa de características como salida. Fuente [Khan et al., 2018].

2.3.1. Capas convolucionales

Las capas convolucionales son el componente más importante de una CNN pues son fundamentales para la extracción de características de un mapa de entrada. Estas capas están formadas por filtros (también conocidos como núcleos o kernels convolucionales) cuya función es convolucionar una entrada determinada y generar un mapa de características como resultado [Li et al., 2021]. Un filtro es una matriz de números discretos, cuyos pesos (valores en las cuadrículas de la matriz) se ajustan durante el entrenamiento de la CNN. Estos pesos se inicializan de manera aleatoria al inicio y luego se ajustan en cada iteración. La capa de convolución realiza una serie de cálculos matemáticos que consisten en una multiplicación entre el filtro y una región de la capa de entrada (con el mismo tamaño del filtro) para poder generar valores destinados al mapa de características que se obtiene como salida de esta capa



(a)



(b)

Figura 2.5: Aplicación de un filtro o núcleo de realce (*emboss*). La imagen en la parte derecha (b) muestra el resultado de la aplicación del filtro de realce de 2×2 a la imagen original (a).

(ver ejemplo en Figura 2.4). Este proceso se realiza de tal modo que el filtro se desliza por pasos a lo largo y alto de la entrada. Tanto las dimensiones del filtro, así como el paso, son determinados previamente [Khan et al., 2018]. La imagen mostrada en la Figura 2.5 permite apreciar la aplicación de un filtro de realce que destaca algunas regiones de la imagen original.

2.3.2. Capas de *Pooling*

Después de la convolución, los mapas de características constan de una gran cantidad de características que tienden a causar problemas de sobreajuste [Hawkins, 2004]. Para abordar este problema, se usan las capas de agrupamiento (también conocidas como *capas de pooling*) . Estas capas operan con una ventana de un tamaño definido, como se aprecia en la Figura 2.6. En este ejemplo, usando una ventana de 2×2 se calcula el máximo de los valores que aparecen en cada región definida por ésta [Khan et al., 2018]. La ventana recorre el mapa de entrada a un paso determinado (de forma similar a la convolución) aplicando una función de agrupamiento como la función promedio o la función máximo (conocida como *Max-Pooling*). El objetivo

de esta función es reducir de manera efectiva el tamaño del mapa de características dando como resultado un mapa de salida con las características más relevantes. Estas capas permiten reducir la cantidad de parámetros y la complejidad computacional de la red [Chollet, 2021].

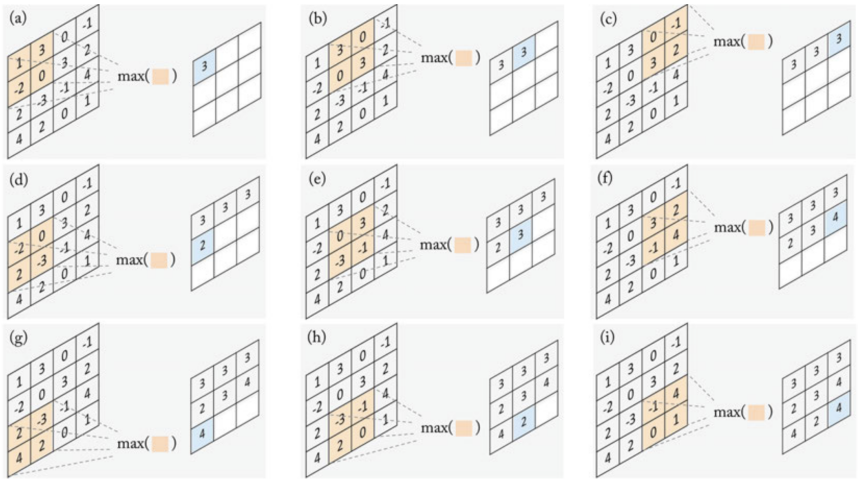


Figura 2.6: Operación **Max-Pooling** aplicada a un mapa de características para reducir su dimensión con los valores máximos (celdas azules) de cada *ventana de muestra* (matriz naranja). Fuente [Khan et al., 2018].

La Figura 2.7 muestra la aplicación de una capa de agrupamiento *max-pooling* a una imagen de 200×260. En la imagen de la derecha se aprecia que las dimensiones de ésta cambian a 100×130, pero la información de la primera se mantiene.

2.3.3. Aplanamiento

El aplanamiento (conocido como *flattening*) es una técnica para convertir una matriz de características multidimensional en un vector unidimensional. Esta técnica se utiliza para conectar las capas de extracción de características con la *capa totalmente conectada*. Para aplanar una matriz de características, simplemente se toma cada fila de la matriz y se coloca en un vector unidimensional [Chollet, 2021].

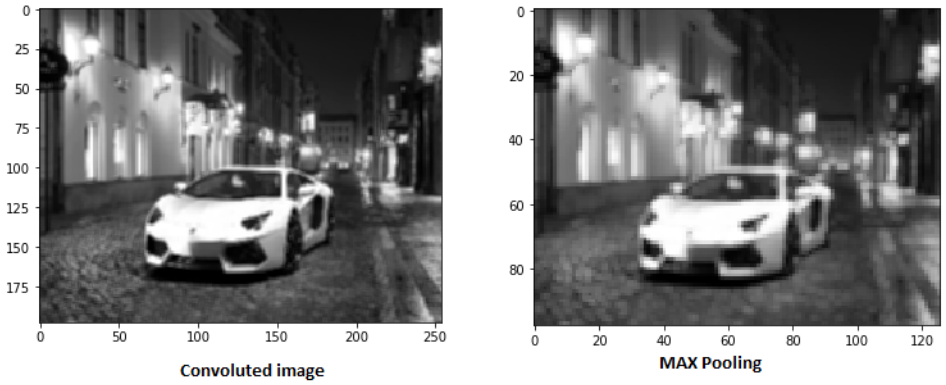


Figura 2.7: Aplicación de max pooling a una imagen. En la imagen original, en la imagen de la izquierda se tiene una imagen resultado de la aplicación de una capa convolucional, en la parte derecha se aprecia la imagen resultado de la capa de *max pooling*.

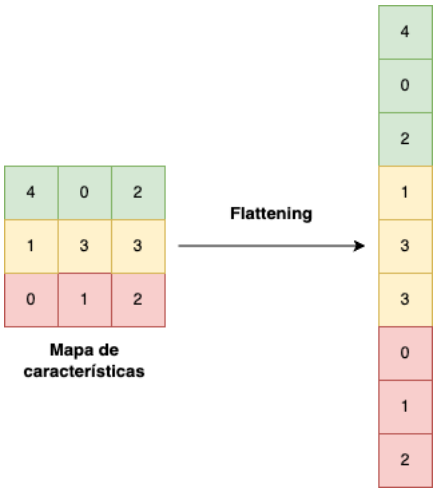


Figura 2.8: Ejemplo de aplanamiento o *"flattening"* a un mapa de características.

2.3.4. Capas totalmente conectadas

Finalmente, el vector de datos unidimensional que se obtuvo en el *aplanamiento* se introduce en un clasificador que no es más que una red neuronal basada en el *Perceptrón Multicapa* [Khan et al., 2018]. Ésta consta de un conjunto de capas o niveles que comprenden una jerarquía de procesamiento. Estos niveles están compuestos por neuronas artificiales, donde cada neurona está conectada directamente tanto a las

neuronas de la capa anterior como a las de la capa siguiente [Albawi et al., 2017]. La Figura 2.9 muestra una representación gráfica de una red totalmente conectada donde se puede apreciar como las características del vector unidimensional generado previamente en el *aplanamiento* se conectan al conjunto de capas ocultas y finalmente se obtiene un vector unidimensional que representa la capa de salida que contiene las predicciones.

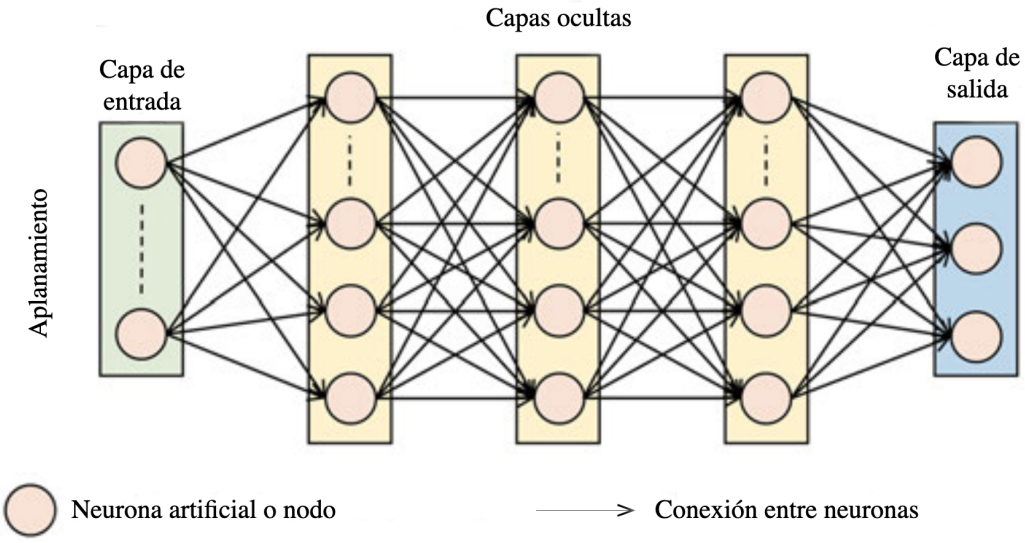


Figura 2.9: Ejemplo de una red neuronal artificial compuesta por la *capa de entrada*, *capas ocultas* y la *capa de salida*. Fuente [Khan et al., 2018].

Los modelos de neuronas artificiales son las unidades de procesamiento de las capas mencionadas. Estas neuronas están asociadas a un valor conocido como *activación* o *salida* que refiere al resultado de una serie de cálculos y operaciones que ocurren en la neurona en base a las entradas que recibe. La activación se calcula mediante una función que toma las entradas ponderadas de cada neurona que representan a un peso que especifica la fuerza de la conexión o relación entre dos nodos [Khan et al., 2018]. El valor de la activación se calcula mediante un modelo matemático conocido como Unidad Lógica de Umbral (TLU, por sus siglas en inglés, *Threshold Logic Unit*), propuesto por [Fitch, 1944] y denotado como:

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

donde y es la salida de la neurona como resultado de sus cálculos, $f(\cdot)$ es una *función de activación*, $\sum_{i=1}^{n} w_i x_i + b$ representa a la suma ponderada de las entradas (x_i) multiplicado por sus respectivos pesos (w_i), más un sesgo (b). La salida de esta función y los valores de las entradas permiten a la red aprender y modelar patrones específicos durante el entrenamiento.

2.3.5. Funciones de activación

Las CNN tienen la capacidad de utilizar diversas funciones de activación para representar características complejas. De manera análoga al funcionamiento de las neuronas en el cerebro humano, la función de activación en una red neuronal determina qué información debe ser transmitida a la siguiente neurona. Tanto las capas de convolucionales como las completamente conectadas van seguidas de una función de activación las cuales determinan por medio de un valor si el nodo se activará o no según la entrada proporcionada. La función toma un valor real dentro de un rango pequeño que suele ser entre $[0, 1]$ o $[-1, 1]$ [Li et al., 2021]. Las funciones de activación introducen no linealidad a la salida de la neurona y permite que la red aprenda relaciones y patrones complejos en los datos [Khan et al., 2018]. A continuación se mencionan unas de las más importantes.

- **Sigmoide:** La función *sigmoide* toma un número real como entrada y regresa un valor en un rango entre $[0, 1]$. Esta función es una de las funciones de activación no lineal más típicas con forma de S como se muestra en la Figura 2.10a (a) [Li et al., 2021]. Esta función está definida en la ecuación 2.2:

$$f_{sigm}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

- **Tanh:** La función *tanh* usa la función de la tangente hiperbólica que regresa un valor en un rango entre $[-1, 1]$. Dado que el valor medio de la salida de *tanh* es 0 como se muestra en la Figura 2.10a (b), se puede lograr una especie de normalización [Li et al., 2021]. Esta función está definida en la ecuación 2.3:

$$f_{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

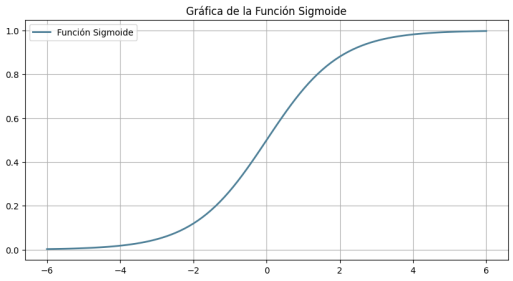
- **Rectifier Linear Unit (ReLU):** $ReLU$ es una función importante debido a su rápido cálculo y se inspira en el procesamiento visual en la corteza humana [Hahnloser et al., 2000]. Esta función determina la salida como 0 si la entrada es negativa y mantiene su valor si es positiva. Esta función está representada por la ecuación 2.4:

$$f_{ReLU}(x) = max(0, x) \tag{2.4}$$

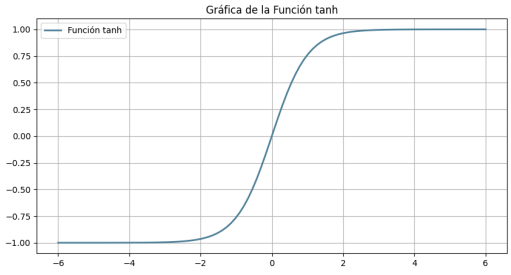
La Figura 2.10 muestra la gráfica de las 3 funciones descritas anteriormente en las ecuaciones 2.2, 2.3 y 2.4.

2.3.6. Optimización basada en gradientes

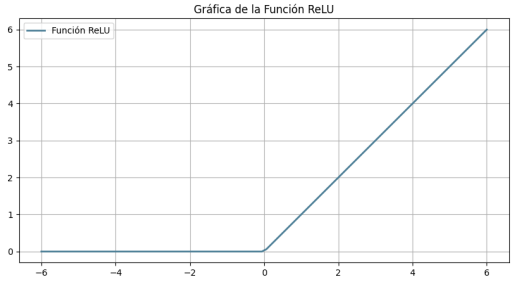
El aprendizaje basado en gradientes en CNN implica ajustar los parámetros de la red para mapear correctamente el espacio de entrada al espacio de salida. Durante el entrenamiento, se compara la estimación actual de las variables de salida con la salida deseada. Esta comparación se utiliza como *función objetivo* en el entrenamiento de la CNN y se conoce comúnmente como *función de pérdida*. En este aspecto se busca ajustar los parámetros de manera adecuada para minimizar la función de pérdida. Estos parámetros incluyen los pesos ajustables de cada capa de la CNN, así como los pesos de los filtros y sesgos de las capas de convolución. Podemos expresar cada iteración de entrenamiento en el momento t utilizando la siguiente ecuación de actualización de parámetros:



(a) Sigmoide



(b) tanh



(c) ReLU

Figura 2.10: Gráficas de funciones de activación.

.

$$\theta_t = \theta_{t-1} - \eta\delta_t \tag{2.5}$$

$$\text{s.t.,} \quad \delta_t = \nabla_\theta \mathcal{F}(\theta_t) \tag{2.6}$$

donde $\mathcal{F}(\cdot)$ representa la función definida por la red neuronal con sus parámetros θ, ∇ representa al gradiente y η se refiere a la tasa de aprendizaje [Khan et al., 2018].

A continuación, se presentarán tres variantes comunes de los métodos de optimización basados en gradientes.

Descenso de Gradiente por Lotes

El *Descenso de Gradiente por Lotes* (del inglés *Batch Gradient Descent*), utiliza todo el conjunto de datos para calcular el gradiente. Aunque es efectivo para problemas convexos, puede ser lento para conjunto de datos grandes como en los problemas de visión por computadora. Por lo tanto, esta técnica puede resultar demasiado lenta debido a que necesita calcular el gradiente en todo el conjunto de datos para cada actualización de parámetros [Ruder, 2016].

Descenso de Gradiente Estocástico

El *Descenso de Gradiente Estocástico* (del inglés "*Stochastic Gradient Decent*")) actualiza los parámetros para cada par de datos de entrenamiento, pero su convergencia puede ser inestable con tasas de aprendizaje altas o con conjuntos de datos diversos. Con una tasa de aprendizaje adecuada, el SGD puede tener un comportamiento de convergencia similar al del *Descenso de Gradiente por Lotes* [Ruder, 2016].

Descenso de Gradiente Mini Batch

El *Descenso de Gradiente Mini Batch* (del inglés *Mini-Batch Gradient Descent*) es una variante del *Descenso de Gradiente Estocástico* que mejora la eficacia y estabilidad de la convergencia del gradiente. Divide el conjunto de entrenamiento en mini-lotes y actualiza los parámetros después de calcular los gradientes en cada mini-lote. Esto resulta en una tasa de convergencia más rápida que el *Descenso de Gradiente Estocástico* y una mayor estabilidad que el *Descenso de Gradiente por Lotes* [Ruder, 2016].

2.3.7. Optimizadores

Cuando se realiza el ajuste de parámetros por medio del descenso de gradiente, es importante tener en cuenta ciertas complicaciones. Establecer la tasa de aprendizaje puede ser complicado y el proceso de entrenamiento puede verse afectado por la inicialización de los parámetros. Además pueden surgir problemas con gradientes que desaparecen o explotan en redes profundas. El entrenamiento puede quedar atrapado en mínimos locales [Khan et al., 2018]. A continuación se mencionan algunos de los métodos que ayudan a tratar este tipo de limitaciones.

Momentum

La optimización con *Momentum* es una mejora del *Descenso de Gradiente Estocástico* que ofrece una convergencia más eficiente. A diferencia del SGD, que puede oscilar cerca de un mínimo local, el método de *Momentum* acelera la convergencia al añadir un coeficiente de momento γ a la ecuación 2.5 en el paso de tiempo anterior a_{t-1} como se muestra en las ecuaciones 2.7 y 2.8:

$$\theta_t = \theta_{t-1} - \alpha_t \tag{2.7}$$

$$\alpha_t = \eta \nabla_\theta \mathcal{F}(\theta_t) + \gamma a_{t-1} \tag{2.8}$$

Momentum acelera la velocidad de convergencia al enfocarse en las direcciones de gradiente consistentes y al mismo tiempo minimiza oscilaciones innecesarias que podrían ralentizar el proceso de entrenamiento [Khan et al., 2018].

Adaptative Gradient

El algoritmo de gradiente adaptativo (AdaGrad, del inglés *Adaptative Gradient*) ajusta la tasa de aprendizaje para cada parámetro i, basándose en su frecuencia en el conjunto de entrenamiento. Esto se logra dividiendo la tasa de aprendizaje de cada parámetro por la acumulación del cuadrado de todos sus gradientes históricos en cada paso de tiempo t para cada parámetro θ_i como se muestra en la ecuación 2.9:

$$\theta_t^i = \theta_{t-1}^i - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} \delta_\tau^{i2} + \epsilon}} \delta_t^i \tag{2.9}$$

donde δ_t^i representa el gradiente en el instante de tiempo t con respecto al parámetro θ_i, y ϵ denota un valor pequeño cercano a 0. Ajustar la tasa de aprendizaje individualmente para cada parámetro elimina la necesidad de establecer manualmente un valor fijo para la tasa de aprendizaje. Usualmente, η se mantiene constante en un solo valor (generalmente entre 10^{-2} o 10^{-3}) durante la fase de entrenamiento [Duchi et al., 2011].

RMSprop

RMSprop propuesto por , es un método que aborda el problema de la disminución de la tasa de aprendizaje en el algoritmo AdaGrad [Khan et al., 2018]. RMSProp calcula el promedio móvil como se aprecia en la ecuacion 2.10:

$$E[\delta^2]_t = \gamma E[\delta^2]_{t-1} + (1-\gamma)\delta_t^2 \tag{2.10}$$

donde γ tiene un valor de 0.9 generalmente. La regla de actualización de los parámetros ajustables queda definida en la ecuación 2.11:

$$\theta_t^i = \theta_{t-1}^i - \frac{\eta}{\sqrt{E[\delta^2]_t + \epsilon}}\delta_t^i \tag{2.11}$$

Estimación del momento adaptativo

El algoritmo de *gradiente adaptativo* (AdaGrad) es útil en situaciones donde los gradientes son dispersos, pero tiene el problema de que la tasa de aprendizaje disminuye con el tiempo. Por otro lado, RMSProp mantiene una tasa de aprendizaje constante en pasos de tiempo altos, pero no es óptimo para gradientes dispersos. La Estimación de Momento Adaptativo (del inglés ADAptative Moment Estimation **ADAM**) combina las ventajas de AdaGrad y RMSprop. Estima una tasa de aprendizaje distinta para cada parámetro y utiliza tanto el primer como segundo momento del gradiente para las actualizaciones. Esto significa que mantiene un promedio móvil de los gradientes (media) y un promedio móvil de los gradientes al cuadrado (varianza) [Kingma and Ba, 2014] como se muestra en las ecuaciones 2.12 y 2.13:

$$E[\delta]_t = \gamma E[\delta]_{t-1} + (1-\gamma_1)\delta_t \tag{2.12}$$
$$E[\delta^2]_t = \gamma E[\delta^2]_{t-1} + (1-\gamma_2)\delta_t^2 \tag{2.13}$$

donde γ_1 y γ_2 son los parámetros que controlan los promedios móviles de la media de la varianza respectivamente. Como las estimaciones del momento se establecen en cero, estos parámetros pueden permanecer en valores muy pequeños incluso después de muchas iteraciones especialmente cuando $\gamma_{1,2} \neq 1$. Para abordar este problema se obtienen estimaciones de $E[\delta]_t$ y $E[\delta^2]_t$ corregidas por el sesgo de inicialización [Kingma and Ba, 2014] mostradas en las ecuaciones 2.14 y 2.15:

$$\hat{E}[\delta]_t = \frac{E[\delta]_t}{1-(\gamma_1)^t} \tag{2.14}$$

$$\hat{E}[\delta^2]_t = \frac{E[\delta^2]_t}{1-(\gamma_2)^t} \tag{2.15}$$

La regla de actualización de los parámetros ajustables para ADAM queda definida en la ecuación 2.16:

$$\theta_t^i = \theta_{t-1}^i - \frac{\eta}{\sqrt{\hat{E}[\delta^2]_t + \epsilon}} \hat{E}[\delta]_t \tag{2.16}$$

Adam, en comparación con otros optimizadores como AdaGrad, RMSprop (mencionados previamente), SGDNesterov o AdaDelta, ha demostrado un rendimiento superior en diversas experimentaciones. En la Figura 2.11, se ilustra cómo el optimizador Adam logró un menor costo durante el entrenamiento con el conjunto de datos MNIST utilizando una Red Neuronal Multicapa con gran diferencia en comparación de los otros optimizadores. Adam suele adaptarse muy bien a problemas de gran escala y exhibe buenas propiedades de convergencia. Es por eso que Adam suele ser la opción predeterminada para muchas aplicaciones de visión por computadora basadas en el aprendizaje profundo [Khan et al., 2018].

2.3.8. Validación cruzada

La validación cruzada (del inglés *Cross Validation*) es una técnica utilizada en el campo del aprendizaje automático y la estadística para evaluar y seleccionar modelos de manera más robusta. Esta técnica se emplea para asegurarse de que un modelo de aprendizaje automático sea capaz de generalizar de manera correcta a datos no vistos y prevenir el sobreajuste.

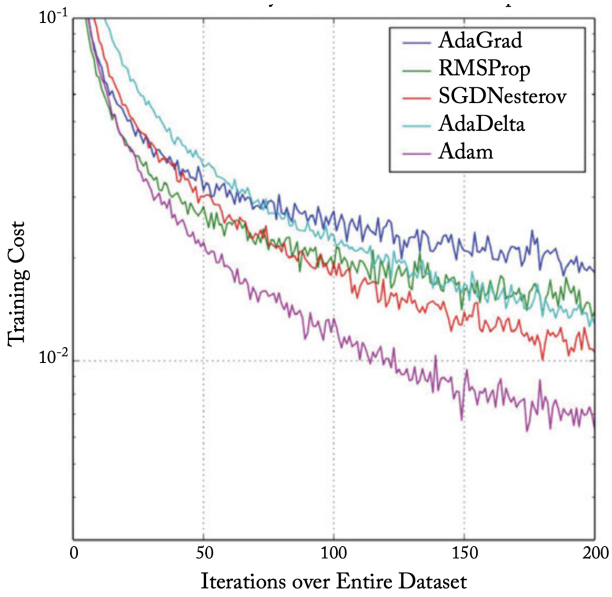


Figura 2.11: Evolución del costo de entrenamiento para seis algoritmos de optimización (AdaGrad, RMSProp, SGD Nesterov, AdaDelta, Adam y SGD) a lo largo de varias iteraciones en todo el conjunto de datos de MNIST. Fuente [Khan et al., 2018].

En la validación cruzada, los datos disponibles se dividen en dos conjuntos principales: un *conjunto de entrenamiento* y un *conjunto de prueba*. Sin embargo, el conjunto de entrenamiento a su vez se divide en varios subconjuntos. El modelo se entrena repetidamente utilizando diferentes combinaciones de subconjuntos de entrenamiento y se evalúa en un conjunto de prueba diferente (también conocido como *conjunto de validación*). Esto permite evaluar el rendimiento del modelo en múltiples conjuntos de datos distintos.

La validación cruzada ayuda a seleccionar el mejor modelo al considerar su rendimiento promedio en diferentes conjuntos de prueba. Esto es esencial para evaluar el sobreajuste, donde un modelo funciona muy bien con los datos de entrenamiento pero no generaliza bien a datos no vistos. Al evaluar el modelo en conjuntos de validación separados, se obtiene una evaluación más sólida de su capacidad para generalizar a nuevos datos [Haykin, 2009].

La validación cruzada presenta varias ventajas en comparación a una sola divi-

sión de los datos en conjuntos de entrenamiento y prueba. Con una única división aleatoria, existe el riesgo de que la precisión en el conjunto de prueba sea demasiado alta o baja, dependiendo de si los ejemplos difíciles de clasificar terminan en el conjunto de entrenamiento o de prueba. En contraste, con la validación cruzada, cada ejemplo aparecerá en el conjunto de entrenamiento y en el conjunto de prueba al menos una vez [Muller and Guido, 2017]. Existen diferentes enfoques para la validación cruzada, algunos de ellos se explican a continuación.

KFold

La versión más común de validación cruzada es *KFold*, donde k es es un número entero (generalmente 5 ó 10) en la cual los datos se dividen en k subconjuntos o pliegues de tamaño aproximadamente igual. Se entrena una secuencia de modelos, y en cada iteración, uno de los pliegues se usa como conjunto de prueba o validación, mientras que el resto se usa como conjunto de entrenamiento. Esto se repite para los k pliegues y se calcula la precisión en cada caso. Al final se obtienen k valores de precisión. Sin embargo, el problema principal de usar *KFold* en algunos casos es que no considera la distribución de clases del conjunto de datos. Esto puede llevar a problemas cuando se trabaja con conjunto de datos desequilibrados, donde una clase tiene más muestras que otra. Existe el riesgo de que uno o más pliegues no contengan representación de la clase minoritaria, lo que puede llevar a evaluaciones erróneas [Muller and Guido, 2017].

KFold estratificado

KFold estratificado (del inglés *KFold Stratified*) es una estrategia de validación cruzada que aborda la limitación de la validación cruzada *KFold* estándar en conjunto de datos desequilibrados en términos de clases. En *KFold* estratificado, los datos se dividen en pliegues de tal manera que la proporción entre las clases sea la misma en cada pliegue que en todo el conjunto de datos asegurando que cada pliegue

tenga una representación proporcional de cada clase [Muller and Guido, 2017]. En la Figura 2.12, se ilustra la diferencia entre el enfoque de *KFold Estratificado* y el *KFold* convencional. En *KFold Estratificado* se asegura que los pliegues de prueba y entrenamiento mantengan aproximadamente la misma proporción de datos por clase. En contraste, el *KFold* clásico puede desequilibrar estas proporciones al seleccionar muestras por pliegues. Por ejemplo, en el pliegue 1, el conjunto de pruebas contiene el 100 % de las muestras de la clase 0, mientras que el conjunto de entrenamiento contiene el 100 % de las muestras de las clases 1 y 2. Esta disparidad en la distribución de clases puede llevar a evaluaciones inexactas y resultados incorrectos en el proceso de evaluación

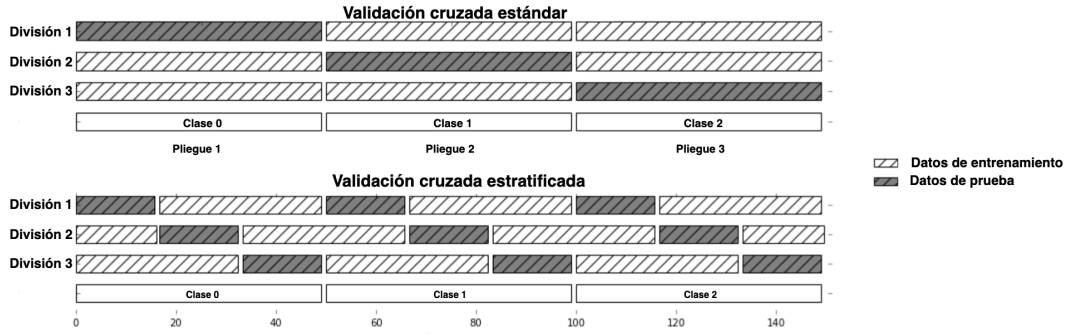


Figura 2.12: Diferencia entre *KFold Estratificado* y *KFold clásico*. Fuente [Muller and Guido, 2017].

2.3.9. Arquitecturas CNN

Considerando los componentes principales de una CNN, es importante destacar la *arquitectura* que refiere a la estructura y módulos que conforman una CNN que está inspirada en la percepción visual.

LeNet-5 es una arquitectura histórica por ser uno de los primeros modelos exitosos en el campo del aprendizaje profundo. VGG-16, por otro lado, es notable por su arquitectura uniforme y profunda que utiliza filtros convolucionales pequeños. Este diseño ha permitido a los investigadores construir modelos más profundos sin un aumento

excesivo en la complejidad, lo que ha llevado a mejoras significativas en la precisión de las tareas de clasificación de imágenes [LeCun et al., 1998]. Además, VGG-16 ha sido ampliamente utilizado como una red preentrenada para transferir aprendizaje, demostrando su versatilidad y robustez [Simonyan and Zisserman, 2014]. A continuación, se explica a profundidad las dos arquitecturas.

Arquitectura LeNet

La arquitectura LeNet es una de las más básicas y populares, creada por Yann LeCun, fue usada originalmente para la clasificación de dígitos realizados a mano en 1989 entrenada por el algoritmo de retropropagación [LeCun et al., 1998]. Se ha utilizado como base para una variedad de aplicaciones de visión por computadora incluyendo la clasificación de imágenes y reconocimiento de objetos. Existe una variante de esta arquitectura llamada *LeNet-5* compuesta por 5 capas de peso en total como se muestra en la Figura 2.13; consta de 2 capas de convolución, cada una seguida por una capa *max-pooling*. Luego una sola capa de convolución seguida de un conjunto de 2 capas totalmente conectadas. Las salidas de estas capas funcionan como clasificador en base a las características extraídas en las capas anteriores. *LeNet-5* es la red neuronal convolucional pionera que combina campos receptivos locales, pesos compartidos y submuestreo espacial o temporal, lo que puede garantizar la invariancia de cambio, escala y distorsión [Li et al., 2021].

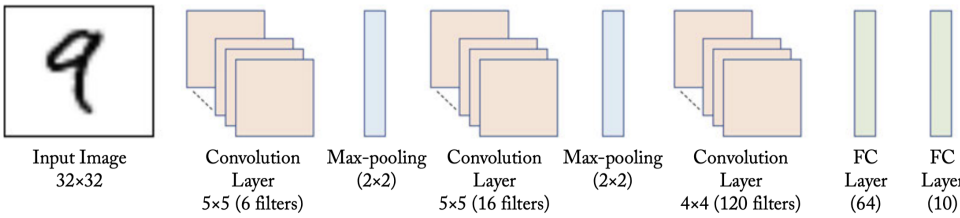


Figura 2.13: Arquitectura LeNet. [Khan et al., 2018].

Arquitectura VGG

La arquitectura VGGNet, presentada en 2014 es uno de los modelos de CNN más populares debido a la simplicidad de su modelo y al uso de núcleos convolucionales de pequeño tamaño, lo que permite crear redes muy profundas. Hay diversas configuraciones de red, siendo las más exitosas la configuración D, conocida como VGGnet-16 [Simonyan and Zisserman, 2014].

VGGnet utiliza núcleos de convolución de tamaño 3×3 con capas intermedias de *maxpooling* para la extracción de características y tres capas completamente conectadas al final para la clasificación. Cada capa de convolución está seguida por una capa $ReLU$ como se muestra en la Figura 2.14. El uso de núcleos más pequeños reduce el número de parámetros, lo que permite un entrenamiento y pruebas eficientes. Al apilar varios núcleos de tamaño 3×3, se puede aumentar el campo receptivo. Esto permite crear redes más profundas, mejorando el rendimiento en tareas de visión artificial [Khan et al., 2018].

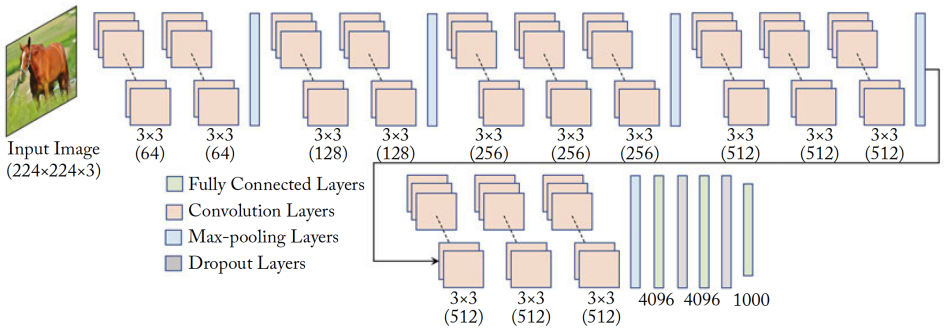


Figura 2.14: Arquitectura *VGG-16*. Fuente[Khan et al., 2018].

2.4. Regularización

Durante el entrenamiento de las CNN se tiene un número muy grande de parámetros a ser ajustado. Normalmente los modelos reaccionan mucho mejor cuando se

presentan los datos de entrenamiento, pero tienden a reducir su rendimiento cuando se presenta el conjunto de prueba, o en otras palabras su capacidad de generalización se puede reducir significativamente cuando se presentan muestras que no ha visto anteriormente. La regularización es un conjunto de técnicas intuitivas que permiten reducir este problema [Khan et al., 2018, Prince, 2023]. Dos de las principales son el aumento de datos y aplicación de *dropout* que se describen a continuación.

2.4.1. Aumento de datos

El aumento de datos es una técnica utilizada en el aprendizaje automático, específicamente en modelos basados en CNN, para mejorar la capacidad de generalización de dichos modelos. Es útil cuando se dispone de un número limitado de muestras de entrenamiento, ya que permite aumentar significativamente el conjunto de datos (generalmente por factores de $16x$, $32x$, $64x$, o más) para entrenar modelos a gran escala [Prince, 2023, Li et al., 2021]. El aumento de datos se logra al crear múltiples copias de una única imagen mediante operaciones y transformaciones aplicadas individualmente o en combinación a la imagen original.

Antes de listar las operaciones más comunes sobre imágenes se define a una imagen digital como una función bidimensional, donde el par ordenado (x, y) representa la ubicación en la imagen y la amplitud de la función en ese punto se conoce como *intensidad*, siendo estos valores finitos y discretos. Las imágenes están compuestas de elementos llamados "pixeles", que son los componentes fundamentales de la representación digital de una imagen [Gonzalez, 2009].

Algunas de las operaciones para aumento de datos se describen a continuación.

Figura 2.15: Representación de una imagen digital como una matriz bidimensional con valores reales que representan a la intensidad de cada píxel. Fuente [Powell, 2023].

2.4.2. Operaciones sobre imágenes

Las operaciones realizadas sobre cada una de las imágenes generan una nueva imagen como resultado de realizar una *operación* sobre la imagen de entrada original. A continuación, se mencionan algunas operaciones comunes utilizadas en el aumento de datos.

Reflexión

La Reflexión consiste en reflejar una imagen sobre el eje X o el eje Y de manera similar a como se vería en un espejo. Cuando se aplica esta operación en el eje X (horizontal), los elementos que estaban en la parte izquierda de la imagen original, pasan a estar en la parte derecha de la imagen reflejada, y viceversa, como se aprecia en la Figura 2.16b(b). Por otro lado, cuando se aplica esta transformación sobre el eje Y (vertical), se reflejan los elementos de la imagen de arriba hacia abajo y viceversa (ver Figura 2.16c(c)) [Shorten and Khoshgoftaar, 2019].

(a) (b) (c)

Figura 2.16: Reflexión aplicada a una imagen. (a) Imagen Original. (b) Imagen espejo horizontal. (c) Imagen espejo vertical. Imagen fuente [Kelly, 2001]

Rotación

La rotación consiste en girar una imagen hacia la derecha o izquierda en un ángulo entre 1° y 359° [Kinser, 2018, Shirley et al., 2009]. Formalmente, se refiere a la rotación de un vector $\overrightarrow{x}$ descrito por:

$$\overrightarrow{y} = \mathbf{R}\overrightarrow{x}, \tag{2.17}$$

donde $\mathbf{R}$ es la matriz de rotación,

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.18}$$

y θ es el ángulo de rotación. Para realizar una rotación en una imagen, se toma la coordenada de cada píxel y se emplea como un componente de la ecuación 2.17 [Kinser, 2018].

(a) (b)

Figura 2.17: Rotación aplicada a una imagen. (a) Imagen Original. (b) Imagen rotada. Imagen fuente [Kelly, 2001]

Acercamiento

Se refiere al proceso de agrandar una región de una imagen. La interpolación es fundamental para lograr el efecto de acercamiento (*zoom*), ya que permite estimar los valores de píxeles desconocidos al aumentar su tamaño [Gonzalez, 2009].

(a) (b)

Figura 2.18: Zoom aplicado a una imágen. (a) Imagen original. (b) Imagen con zoom. Imagen fuente [Kelly, 2001]

Brillo

Se refiere a la percepción subjetiva de la intensidad de una imagen o de los valores discretos que representan la intensidad de la luz en una imagen. Es la forma en que percibimos cuán claro u oscuro es un punto en específico de la imagen [Gonzalez, 2009].

(a) (b) (c)

Figura 2.19: Modificación del brillo en una imagen. (a) Imagen original. (b) Imagen con brillo aumentado. (c) Imagen con brillo reducido. Imagen fuente [Kelly, 2001]

2.4.3. Dropout

Una de las técnicas de regularización más utilizadas que ayuda a reducir el sobre ajuste en redes neuronales es el método conocido como "*dropout*". Su aplicación se centra en el proceso de entrenamiento de la red, donde cada neurona se activa con una probabilidad fija, comúnmente establecida en 0.5, o adaptada mediante un conjunto de validación. En cada iteración del entrenamiento, se selecciona de forma aleatoria un subconjunto de neuronas activas, lo que genera una especie de "subred"dentro de la red global, generando así un efecto de "conjunto"durante la fase de prueba.

El *dropout* funciona desactivando aleatoriamente una fracción de las neuronas de la red durante el entrenamiento [Li et al., 2021]. Esto obliga a la red a aprender a utilizar un conjunto más diverso de neuronas, lo que la hace más robusta y generalizable. Esta técnica es altamente efectiva para la regularización y conduce a un aumento significativo en el rendimiento en datos no vistos en la fase de prueba,

lo que resulta en un modelo más sólido y con mayor capacidad de generalización [Khan et al., 2018].

2.4.4. Transferencia de Aprendizaje

La transferencia de aprendizaje es una técnica que ayuda a prevenir el sobreajuste en los modelos CNN. La transferencia de aprendizaje implica entrenar una red con un conjunto de datos grande, como *ImageNet* [Deng et al., 2009] , y luego utilizar los pesos obtenidos de este entrenamiento previo como pesos iniciales para nuevas tareas de clasificación. Comúnmente, sólo se copian los pesos de las capas convolucionales, en lugar de toda la red, que incluye las capas totalmente conectadas. Esta estrategia es efectiva ya que muchas imágenes comparten características espaciales de alto nivel que se aprenden mejor con grandes conjuntos de datos. Al hacerlo, se logra una representación más rica y diversa de las características, lo que mejora la capacidad del modelo para generalizar a partir de nuevos datos [Zamir et al., 2018].

2.5. Métricas

2.5.1. Matriz de confusión

Sea $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ el conjunto de clases (etiquetas) reales correspondientes al conjunto de datos D, donde α_i es la clase real del elemento d_i. Sea $E = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$ el conjunto de clases estimadas por el clasificador C de cada elemento en D, donde ε_i es la clase estimada del elemento d_i. El rendimiento de un clasificador C puede ser evaluado mediante una función M que asigna una métrica $\mu \in \mathbb{R}$ al par ordenado (A, E), que es $(A, E) \xrightarrow{M} \mu$. De esta manera, por definición una matriz de confusión $C_{i,j}$ donde todos los valores $m_{i,j}$ indican la cantidad de aciertos y errores otorgados por el modelo evaluado en cuestión, en donde cada fila i corresponde a los

valores reales de las clases de un conjunto de elementos, mientras que las columnas j representan a los valores predichos de las clases por un modelo [Luque et al., 2019].

$$C_{i,j} = \begin{pmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{pmatrix} \tag{2.19}$$

Para clasificación binaria, los elementos de la matriz se denominan como clase "positiva" y clase "negativa", por lo tanto la matriz de confusión se puede representar como:

$$C_{i,j} = \begin{pmatrix} m_{TP} & m_{FN} \\ m_{FP} & m_{TN} \end{pmatrix} \tag{2.20}$$

Donde:

- TP (Verdadero positivo): corresponde a los casos en los que el modelo ha predicho correctamente una clase positiva.
- TN (Verdadero negativo): corresponde a los casos en los que el modelo ha predicho correctamente una clase negativa.
- FP (Falso positivo): corresponde a los casos en los que el modelo ha predicho incorrectamente una clase positiva.
- FN (Falso negativo): corresponde a los casos en los que el modelo ha predicho incorrectamente una clase negativa.

En la matriz de confusión se busca que la matriz diagonal presente la mayor concentración de números que representan a los valores de los verdaderos positivos y los verdaderos negativos. Gracias a estos elementos en la matriz de confusión, se pueden calcular diversas métricas, como son: precisión, *recall*, *F1-score*, coeficiente

de correlación de Matthews, sensitividad, especificidad, entre otras. La matriz de confusión es una herramienta valiosa para entender la eficacia de un modelo de clasificación.

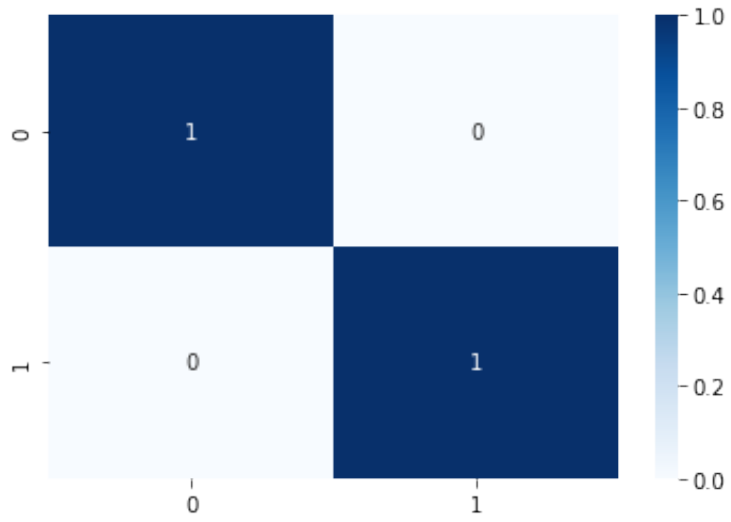


Figura 2.20: Ejemplo de matriz de confusión

2.5.2. Exactitud

La exactitud (*accuracy* en inglés) es una métrica de rendimiento utilizada para evaluar el desempeño de un modelo de clasificación. La exactitud se define como la relación de predicciones correctas que hace el modelo en comparación con el número total de predicciones hechas [Luque et al., 2019].

A partir de los valores obtenidos por la matriz de confusión, la exactitud se puede calcular de la siguiente forma:

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

2.5.3. Sensitividad

La Sensitividad se refiere a la capacidad del modelo para identificar correctamente los verdaderos positivos. Es la proporción de positivos reales que se identifican correctamente [Luque et al., 2019]. Se calcula como:

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

2.5.4. Especificidad

La especificidad es una métrica que mide la capacidad de un modelo para identificar correctamente los verdaderos negativos. Es la proporción de negativos reales que se identifican correctamente [Luque et al., 2019]. Se calcula como:

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

2.5.5. Precisión

La precisión es una métrica que calcula la exactitud de las predicciones con respecto a las etiquetas. Es una operación idempotente que simplemente divide los verdaderos positivos por la suma de verdaderos positivos y falsos positivos. Se calcula como [Luque et al., 2019]:

$$Precision = \frac{TP}{TP+FP}$$

2.5.6. F1-Score

La puntuación F (del inglés F1-Score) también conocida como F balanceada o F media, es una métrica que combina la precisión y el *recall* en una sola medida. Es útil cuando la distribución de clases en un conjunto de datos es desequilibrada y se requiere un equilibrio entre la capacidad de un modelo para identificar correctamente las clases positivas y evitar falsos positivos [Luque et al., 2019]. El F1-Score se calcula como la media armónica entre la precisión y el recall tal que:

$$F_1 Score = 2 \cdot \frac{PRC \cdot SNS}{PRC + SNS}$$

Donde PRC es Precisión y SNS Sensitividad.

2.5.7. Coeficiente de correlación de Mattews

El coeficiente de correlación de Matthews (MCC, por sus siglas en inglés) es una medida estadística que se utiliza para evaluar la calidad de las predicciones de un modelo en comparación con los datos reales. El MCC es una medida de correlación entre los datos observados y los datos predichos. Tenemos que $MCC \in [-1, 1]$, donde 1 nos indica una correlación perfecta, 0 nos indica una falta de correlación y un -1 indica una correlación inversa perfecta [Luque et al., 2019]. Así pues, el MCC puede ser calculado como:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

2.6. Bibliotecas de desarrollo

Las bibliotecas de desarrollo tienen un papel fundamental al proporcionar las herramientas necesarias para crear aplicaciones de manera eficiente, que además facilitan la implementación de funcionalidades complejas ayudando a acelerar el proceso de desarrollo.

2.6.1. TensorFlow

TensorFlow es una de las bibliotecas de aprendizaje profundo más populares y utilizadas. *TensorFlow* está escrito sobre un motor *C/C++* capaz de ejecutar cálculos numéricos de gran complejidad a una alta velocidad [Khan et al., 2018]. *TensorFlow* brinda potentes herramientas para implementar modelos de ML en cualquier entorno, como servidores, dispositivos móviles, navegadores, microcontroladores, CPU, GPU o FPGA [TensorFlow, 2023a].

2.6.2. TensorFlow.js

TensorFlow.js es una herramienta poderosa que amplía las capacidades de *TensorFlow* hacia el desarrollo de aplicaciones de aprendizaje automático en entornos web y móviles, utilizando *JavaScript* o *TypeScript*. Esta extensión permite a los desarrolladores trabajar con frameworks populares como *Node.js*, *React* y *Angular*, brindando así flexibilidad y compatibilidad con una amplia gama de plataformas. Además, facilita el despliegue de modelos de ML tanto en navegadores como en dispositivos móviles, lo que abre un mundo de posibilidades para la creación de aplicaciones inteligentes y accesibles en diferentes contextos.

Una característica destacada de *TensorFlow.js* es su capacidad para crear modelos directamente en *JavaScript*, aprovechando las herramientas y funcionalidades

de TensorFlow. No obstante, también ofrece la posibilidad de importar modelos previamente desarrollados en *Python* con *Keras*, lo que simplifica la integración de modelos existentes y permite aprovechar la versatilidad y potencia de esta biblioteca [TensorFlow, 2023a].

Además, la documentación de *TensorFlow.js* brinda una amplia gama de implementaciones de modelos de aprendizaje automático en una variedad de campos, abarcando desde la clasificación de imágenes hasta la detección de objetos, el reconocimiento de voz, la segmentación y la clasificación de texto, entre otras funcionalidades [TensorFlow, 2024a].

2.6.3. Keras

Keras es una biblioteca de alto nivel enfocado en el aprendizaje profundo de código abierto escrita en Python con capacidad de ejecutarse sobre **TensorFlow**, lo cual aprovecha para realizar tareas complejas de éste último en módulos más simples [Khan et al., 2018]. Se utiliza para la creación y entrenamiento de modelos de *aprendizaje profundo*, como *RNA's*, *CNN*, modelos de regresión lineal, entre otros. *Keras* es ideal para desarrolladores de todos los niveles, ya que ofrece una amplia variedad de funciones que simplifican el desarrollo de estos modelos en unas pocas líneas de código, lo que resulta en un código más limpio y elegante [Keras, 2023].

2.6.4. React Native

React Native es un framework de *JavaScript* que se renderiza con código nativo, que permite la creación de aplicaciones móviles para *Android* y *iOS*. *React Native* combina las mejores partes del desarrollo nativo con *React*, una biblioteca de *JavaScript* que permite crear interfaces de usuario de manera rápida y eficiente [Native, 2023a].

Capítulo 3

Desarrollo del modelo CNN

El problema de desarrollar un modelo de redes neuronales convolucionales para un conjunto de datos específico plantea un desafío significativo en el campo del aprendizaje profundo. Las CNN son particularmente eficaces para tareas de procesamiento de imágenes debido a su capacidad para capturar patrones espaciales y jerárquicos en los datos visuales. El primer paso en este proceso es la identificación o generación de un conjunto de datos adecuado que refleje las características y variabilidades necesarias para entrenar el modelo de acuerdo al problema en cuestión. En el caso de la generación de un conjunto de datos a medida, éste debe ser cuidadosamente diseñado y preprocesado para asegurar que el modelo CNN pueda aprender de manera eficiente y generalizar correctamente a nuevos datos. La calidad y la representatividad del conjunto de datos influyen directamente en el rendimiento y la precisión del modelo.

En el presente capítulo se detallan los aspectos más importantes para el desarrollo del modelo CNN. En la Sección 3.1, se abordan las especificaciones tanto del hardware como del software. La Sección 3.2 presenta detalles del conjunto de datos obtenido. En la Sección 3.3, se explica la elección de la arquitectura CNN para los experimentos posteriores. La Sección 3.2.2 discute el proceso de aumento de datos y la edición de

las imágenes que lo conforman, mientras que la Sección 3.3.1 detalla el proceso de búsqueda de los mejores hiperparámetros para la arquitectura CNN. Finalmente, la Sección 3.3.2 describe cómo se utilizó la validación cruzada para refinar el modelo final.

3.1. Especificaciones de hardware y software

El entrenamiento, pruebas, ajuste de modelos de ML se llevaron a cabo en una computadora *MacBook Air M1 2020*. Las características de la computadora se detallan en la Tabla 3.1.

Característica	*MacBook Air M1 2020*
Memoria RAM	*8 GB*
Procesador	*Apple Silicon M1*
GPU	*7 núcleos (M1 - GPU)*
Sistema operativo	*macOS Sonoma 14.0*

Tabla 3.1: Características del equipo de cómputo usado durante el desarrollo.

Los módulos desarrollados en este proyecto fueron implementados en el lenguaje de programación *Python* versión 3.9.13, con la ayuda de las siguientes bibliotecas:

- *Scikit-Learn* versión **1.1.1**
- *Keras* versión **2.13.1**
- *TensorFlow* versión **2.13.0**
- *Seaborn* versión **0.11.2**
- *NumPy* versión **1.21.5**

- *Matplotlib* versión **3.5.2**
- *Pandas* versión **1.5.3**
- *Wandb* versión **1.5.3**

Además se utilizó *GIMP* versión **2.10** para la edición y recorte de las imágenes que conformaron el conjunto de datos, *Weight y Biases* para monitorear los hiperparámetros del modelo durante su entrenamiento y *Visual Studio Code* versión **1.85.0** como entorno de desarrollo.

3.2. Conjunto de datos

El conjunto de imágenes de tilapias del Nilo (*Oreochromis niloticus*) fue obtenido en varias sesiones por personal de la Universidad del Mar (UMAR) campus Puerto Ángel, Oaxaca, México. Para la adquisición de las imágenes se utilizaron cámaras de teléfonos celulares con la intención de tener datos en las mismas condiciones de trabajo de campo en las que se usarán los modelos generados en este trabajo.

La Tabla 3.2 presenta la distribución de las imágenes capturadas que conforman el conjunto de datos. Las imágenes se clasifican por la fecha de captura, el dispositivo móvil utilizado, el género de los ejemplares (macho o hembra), y la cantidad de imágenes obtenidas. Se puede observar que las imágenes fueron capturadas en diferentes fechas utilizando varios modelos de dispositivos móviles.

En la Tabla 3.3 muestra el total de imágenes clasificadas por sexo en el conjunto final en la que se observa que hay un total de 3526 imágenes, con una distribución casi igual entre machos y hembras. Para la validación de los modelos, el conjunto de datos se dividió en dos partes: un conjunto de prueba compuesto por 120 imágenes

para cada género y un conjunto de entrenamiento con las imágenes restantes. De esta forma, el conjunto de entrenamiento está compuesto por 1627 imágenes de machos y 1659 de hembras, como se detalla en la Tabla 3.4.

Fecha de captura	**Dispositivo móvil**	**Género**	**Cantidad**
14-07-2022	*Redmi 9A*	Machos	236
14-07-2022	*Redmi 9A*	Hembras	240
14-07-2022	*Samsung Galaxy A30*	Machos	235
14-07-2022	*Samsung Galaxy A30*	Hembras	240
14-07-2022	*Xiaomi 11*	Hembras	85
15-07-2022	*Huawei Y7a*	Machos	97
15-07-2022	*Huawei Y7a*	Hembras	154
15-07-2022	*Motorola One Fusion*	Machos	141
30-08-2022	*Moto G50*	Machos	196
30-08-2022	*Motorola One Fusion*	Machos	256
30-08-2022	*Poco F3*	Machos	320
30-08-2022	*Redmi Note 9s*	Machos	266
30-08-2022	*Moto G50*	Hembras	265
30-08-2022	*Motorola One Fusion*	Hembras	265
30-08-2022	*Poco F3*	Hembras	266
30-08-2022	*Redmi Note 9s*	Hembras	264

Tabla 3.2: Dispositivos móviles y distribución por sexo de las imágenes que conforman el conjunto de datos.

	Machos	**Hembras**	**Total**
Total	1747	1779	**3526**

Tabla 3.3: Total de imágenes clasificadas por sexo en el conjunto de datos.

	Machos	**Hembras**	**Total**
Entrenamiento	1627	1659	**3286**
Prueba	120	120	**240**

Tabla 3.4: Distribución de imágenes por sexo en los conjuntos de entrenamiento y prueba.

3.2.1. Edición de imágenes

Durante las pruebas iniciales, los modelos que se entrenaron utilizando las imágenes originales mostraron un rendimiento deficiente. Esto se debió a la presencia de contenido adicional en las imágenes que no estaba relacionado con las papilas genitales, como manos y guantes. Para optimizar el conjunto de datos para la experimentación, se tomó la decisión de recortar manualmente las imágenes utilizando el editor de imágenes *GIMP*.

Este proceso permitió la eliminación de cualquier contenido innecesario, centrando las imágenes exclusivamente en el área que contiene las papilas genitales. Como resultado, se obtuvo un conjunto de datos más limpio que permitió a la CNN aprender de manera más precisa y efectiva las características relevantes.

La Figura 3.1 muestra un ejemplo de este proceso de recorte. En la Figura 3.1a, se puede observar la imagen original de una papila genital de tilapia, donde se pueden ver elementos adicionales como las manos del experto y una cubeta de fondo. En la Figura 3.1b, se muestra la misma imagen después del proceso de recorte, donde todo el contenido innecesario ha sido eliminado y la imagen se ha centrado exclusivamente en la papila genital. Este enfoque de recorte resultó en un conjunto de datos más enfocado, lo que facilitó el aprendizaje efectivo de la CNN sobre las características relevantes de las papilas genitales.

Durante recorte manual de las imágenes, se identificó aquellas que no eran aptas para formar parte del conjunto de datos debido a problemas como el desenfoque o la mala calidad. Estos problemas podrían haber introducido ruido y ambigüedad en el conjunto de datos, lo que habría afectado negativamente en el rendimiento de la CNN. Después de esta fase de preprocesamiento el conjunto de imágenes quedó conformado como se muestra en la Tabla 3.5.

(a) (b)

Figura 3.1: Proceso de recorte de una imagen de papila genital de tilapia. (a) Imagen original. (b) Imagen recortada.

	Machos	**Hembras**	**Total**
Aceptadas	1678	1672	**3350**
Descartadas	69	107	**176**

Tabla 3.5: Total de imágenes aceptadas y descartadas después del proceso de recorte y eliminación de las imágenes.

3.2.2. Aumento de datos

Como se mencionó en el capítulo anterior, el aumento de datos tiene por objetivo el incrementar la cantidad de imágenes a usar durante el entrenamiento de una CNN para obtener un modelo más robusto. Para llevar a cabo el aumento de datos, se emplearon varias operaciones de transformación con el fin de ampliar el conjunto de datos. Estas transformaciones incluyen el redimensionamiento, la rotación, reflejo horizontal y vertical, el ajuste del brillo, el acercamiento o *zoom* y las deformaciones de cizallamiento. Para implementar estas operaciones, se utilizó la biblioteca ***Image Data Generator*** de *TensorFlow v2.13.0*. El Código 3.1 muestra el uso de las operaciones mencionadas.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    brightness_range=[0.5,1.0],
    rotation_range=60,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=[0.3, 1.5],
    shear_range=10.0,
)

training_set = train_datagen.flow_from_directory(
    'dataset/training_set',
    target_size=(64, 64),
    batch_size=32,
    shuffle=True,
    class_mode='binary')
```

Código 3.1: Valores aplicados al aumento de datos.

- ***Rescale:*** Escala los valores de píxeles de las imágenes para que estén en el rango de 0 a 1.
- ***Brightness range:*** Aplica cambios aleatorios en el brillo a las imágenes, donde el valor de brillo estará en el rango de 0.5 a 1.0.
- ***Rotation range:*** Realiza rotaciones aleatorias en las imágenes en un rango de hasta 60 grados.
- ***Horizontal flip:*** Realiza inversiones horizontales aleatorias en las imágenes.
- ***Vertical flip :*** Realiza inversiones verticales aleatorias en las imágenes.
- ***Zoom range :*** Aplica acercamiento aleatorio a las imágenes en un rango de 0.3 a 1.5
- ***Shear Range:*** Aplica deformaciones de cizallamiento aleatorias en las imágenes

Así también se configuraron los siguientes parámetros: ***target_size*** que especifica las dimensiones a las que se redimensiona todas las imágenes, en este caso a **(64, 64)** píxeles. El ***batch_size*** que determina cuántas imágenes se cargarán a la vez durante el entrenamiento o evaluación del modelo, siendo **32** el tamaño de cada lote. Finalmente, el parámetro ***class_mode*** se establece en ***'binary'***, indicando que se está abordando un problema de clasificación binaria [TensorFlow, 2024c]. De esta manera, con las operaciones definidas en el Código 3.1, se obtuvieron imágenes como las que se muestran en la Figura 3.2 .

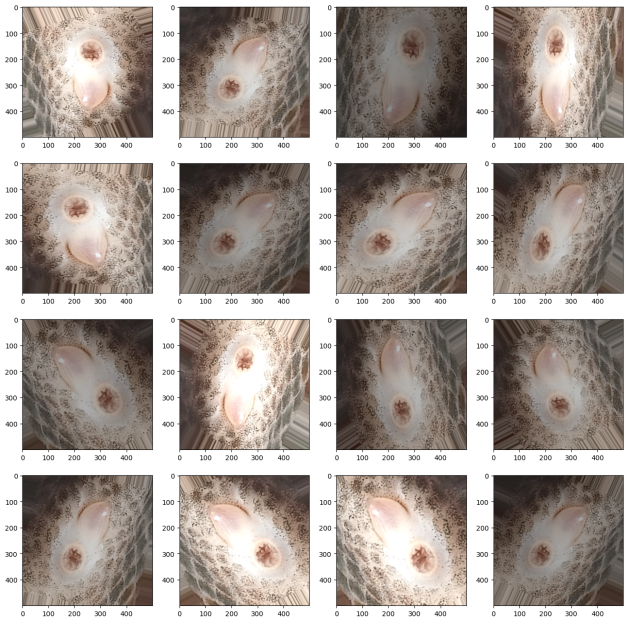


Figura 3.2: Resultados al combinar todas las operaciones definidas en el código para generar distintas transformaciones de una misma imagen con *Image Data Generator*.

3.3. Selección y ajuste de la arquitectura CNN

Siguiendo lo establecido en la metodología, se realizaron una serie de experimentos para seleccionar la arquitectura CNN más adecuada. Se evaluaron dos arquitecturas *VGG-16* y *LeNet-5* [Khan et al., 2018].

Estos experimentos consistieron en el entrenamiento de 5 modelos $M_{1A}, M_{2A}, ..., M_{5A}$ correspondientes a la arquitectura *LeNet-5* y de 5 modelos $M_{1B}, M_{2B}, ..., M_{5B}$ correspondientes a la arquitectura *VGG-16*. Estos 10 modelos resultantes fueron entrenados para poder realizar un análisis del comportamiento de ambas arquitecturas en el conjunto de datos.

El entrenamiento de los modelos CNN se llevó a cabo mediante la librería Keras, permitiendo la configuración manual de los modelos y la adaptación de parámetros específicos para la clasificación binaria. Para aprovechar el conocimiento previamente adquirido en grandes conjuntos de datos como *ImageNet*, se optó por emplear un modelo pre-entrenado disponible en *Keras Applications*, específicamente la arquitectura *VGG-16* [Keras, 2024]. Esta decisión se basó en la estrategia de transferencia de aprendizaje (*transfer learning*), la cual permite utilizar modelos entrenados en un dominio o conjunto de datos para un nuevo escenario, lo que puede resultar en una mejora significativa en la precisión de la clasificación al aprovechar la información aprendida de millones de imágenes. Además, se aplicó *fine-tuning* para ajustar aún más el modelo a las necesidades específicas del problema de clasificación en cuestión [Pan and Yang, 2009].

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 64, 64, 64)	1792
max_pooling2d (MaxPooling2D)	(None, 31, 31, 64)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	18464
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 512)	590336
dropout (Dropout)	(None, 512)	0
dense_9 (Dense)	(None, 1)	513

Total params: 620353
Trainable params: 620353
Non-trainable params: 0

Tabla 3.3: Parámetros elegidos para la experimentación con la arquitectura *LeNet-5*.

La arquitectura y los parámetros definidos para la red *LeNet-5* se presentan en la Tabla 3.3, mientras que su representación visual se muestra en la Figura 3.4. Por otra parte, los parámetros definidos para la arquitectura *VGG-16* se muestran en la

Tabla 3.5 y visualmente se observan en la Figura 3.6.

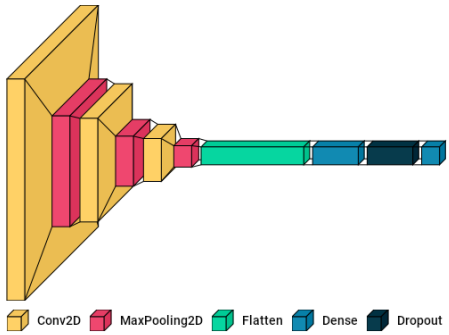


Figura 3.4: Arquitectura de la red *LeNet-5* usada en la experimentación.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 64, 64, 3)]	0
block1_conv1 (Conv2D)	(None, 64, 64, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 64, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 32, 64)	0
block2_conv1 (Conv2D)	(None, 32, 32, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 32, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 16, 128)	0
block3_conv1 (Conv2D)	(None, 16, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 16, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 16, 256)	590080
block3_pool (MaxPooling2D)	(None, 8, 8, 256)	0
block4_conv1 (Conv2D)	(None, 8, 8, 512)	1180160
block4_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block4_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block4_pool (MaxPooling2D)	(None, 4, 4, 512)	0
block5_conv1 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block5_pool (MaxPooling2D)	(None, 2, 2, 512)	0
global_max_pooling2d_1 (GlobalMaxPooling2D)	(None, 512)	0
dense_4 (Dense)	(None, 512)	262656
dropout_2 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 1)	513

Total params: 14977857
Trainable params: 7342593
Non-trainable params: 7635264

Tabla 3.5: Parámetros de la arquitectura *VGG-16* elegidos para la experimentación.

La Tabla 3.6 enlista las funciones utilizadas y más importantes en esta experimentación y proporciona también una descripción de cada una de ellas.

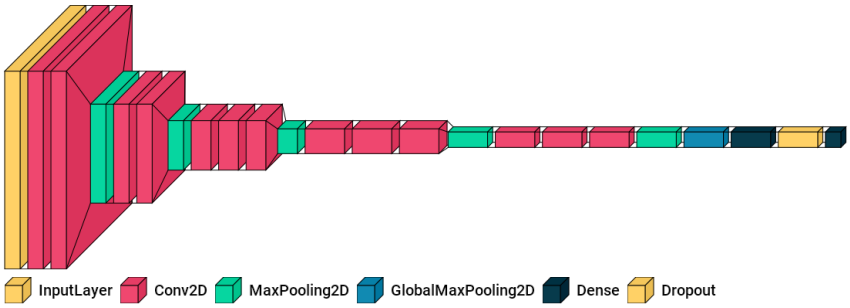


Figura 3.6: Arquitectura *VGG-16* usada en la experimentación.

`def loadData(path):` Este módulo carga el conjunto de datos a través de una ruta especificada. Posteriormente aplica aumento de datos y los devuelve para ser usados en el entrenamiento y pruebas.
`def createLeNet5(input_shape=(64, 64, 3), output_shape=1):` Este módulo crea y devuelve una instancia de un modelo utilizando la arquitectura LeNet-5 con la opción de especificar la forma de entrada y de salida, en este caso para una clasificación binaria.
`def createVGG16(input_shape=(64, 64, 3), output_shape=1):` Este módulo crea y devuelve una instancia de un modelo utilizando la arquitectura *VGG-16* con la opción de especificar la forma de entrada y de salida, en este caso para una clasificación binaria.
`def trainModelVGG16(totalModel = 5):` Este módulo inicia el entrenamiento de múltiples modelos basados en la arquitectura *VGG-16* y guarda cada uno de ellos en formato h5.
`def trainModelLeNet5(totalModels = 5):` Este módulo inicia el entrenamiento de múltiples modelos basados en la arquitectura LeNet-5 y guarda cada uno de ellos en formato h5.
`def predictByBatch(model, pathBatch, classValue, classes):` Este módulo toma un modelo y evalúa un conjunto de datos utilizando la ruta especificada, teniendo en cuenta los valores de las clases proporcionadas como parámetros.
`def testModels(test_set):` Este módulo carga todos los modelos guardados en formato h5 presentes en la carpeta raíz, los evalúa con el lote de pruebas pasado como parámetro y muestra diversas métricas, incluyendo el reporte de clasificación y la matriz de confusión.

Tabla 3.6: Principales módulos utilizados para la elección de una arquitectura CNN.

3.3.1. Ajuste de hiperparámetros.

Después de definir la arquitectura CNN, se procede al ajuste de hiperparámetros con el objetivo de mejorar la capacidad de generalización de la CNN. Este proceso implica la búsqueda de los parámetros óptimos durante el entrenamiento, lo que incluye probar con diferentes configuraciones para mejorar el rendimiento y la precisión del modelo resultante. A continuación, en la Tabla 3.7, se presentan los valores que se probaron durante las experimentaciones y en la Tabla 3.8 muestra los módulos desarrollados para esta experimentación.

Hiperparámetro	**Descripción**	**Valores**
Learning Rate	Es la velocidad a la que un modelo de ML actualiza los pesos mientras aprende de los datos. Un valor bajo puede llevar a convergencia lenta, mientras que uno alto puede resultar en oscilaciones alrededor del mínimo.	0.0005, 0.0025
Dropdout Value	Es un parámetro discreto que se refiere a la fracción de neuronas que se apagan o se excluyen aleatoriamente durante cada paso de entrenamiento [Srivastava et al., 2014]. Esta técnica ayuda a prevenir el sobreajuste en modelos de redes neuronales al evitar que ciertas neuronas dependan demasiado de otras.	0.3, 0.4, 0.5
Epochs	Representa la cantidad de veces que el modelo recorre todo el conjunto de datos durante el entrenamiento. Un mayor número de épocas puede permitir que el modelo aprenda más, pero un exceso puede conducir al sobreajuste.	80, 100, 120
Units	Se refiere a la cantidad de neuronas en la capa totalmente conectada de la red neuronal. El número de unidades puede afectar la complejidad y capacidad de representación del modelo, influyendo en su capacidad para aprender patrones.	512, 1024

Tabla 3.7: Parámetros probados durante el entrenamiento del modelo.

`def loadData(path):` Este módulo carga el conjunto de datos a través de una ruta especificada. Posteriormente aplica aumento de datos y los devuelve para ser usados en el entrenamiento y pruebas.
`def hyperParametersTuning(hyperparameters):` Este módulo realiza un ajuste de hiperparámetros en un modelo CNN mostrando las métricas de cada configuración del modelo evaluado a través de los parámetros proporcionados.
`def createModel(input_shape=(64, 64, 3), output_shape=1):` Este módulo crea un modelo CNN por cada configuración de parámetros con una entrada y salida establecida.
`def saveBestParameters(parameters):` Este módulo muestra los hiperparámetros utilizados para cada configuración y compara su desempeño para guardar los mejores hiperparámetros basados en la precisión obtenida.

Tabla 3.8: Módulos utilizados para el ajuste de hiperparámetros.

3.3.2. Validación cruzada

Una vez obtenidos los hiperparámetros óptimos para la arquitectura CNN, se procedió a la fase final para garantizar una evaluación robusta y precisa del modelo. Esta fase implicó la realización de un proceso de experimentación utilizando 10 pliegues (*folds*), en los que se entrenó un solo modelo durante 15 épocas por cada pliegue.

En cada pliegue de la validación cruzada estratificada, se utilizaron 2799 elementos para el entrenamiento, 311 elementos para la validación y 240 elementos para la prueba. Esta distribución de los datos asegura que el modelo se entrena con una gran cantidad de datos, se valida con una cantidad suficiente para hacer ajustes y finalmente se prueba con datos no mostrados para evaluar su rendimiento.

Siguiendo un enfoque similar al aplicado en trabajos previos [Dekanovský, 2020], se implementó la validación cruzada con un modelo persistente. Como se detalla en el diagrama de la Figura 3.7, después de cada entrenamiento por pliegue, se guardaron las métricas de entrenamiento y el modelo con los pesos de entrenamiento de cada

fold.

Al finalizar, se seleccionó el modelo con el mejor rendimiento basándose en las métricas. Este modelo seleccionado se utilizó finalmente dentro de la aplicación móvil. El modelo se configuró con los hiperparámetros óptimos obtenidos en la fase anterior. Para estos experimentos, se utilizaron los módulos de la Tabla 3.9.

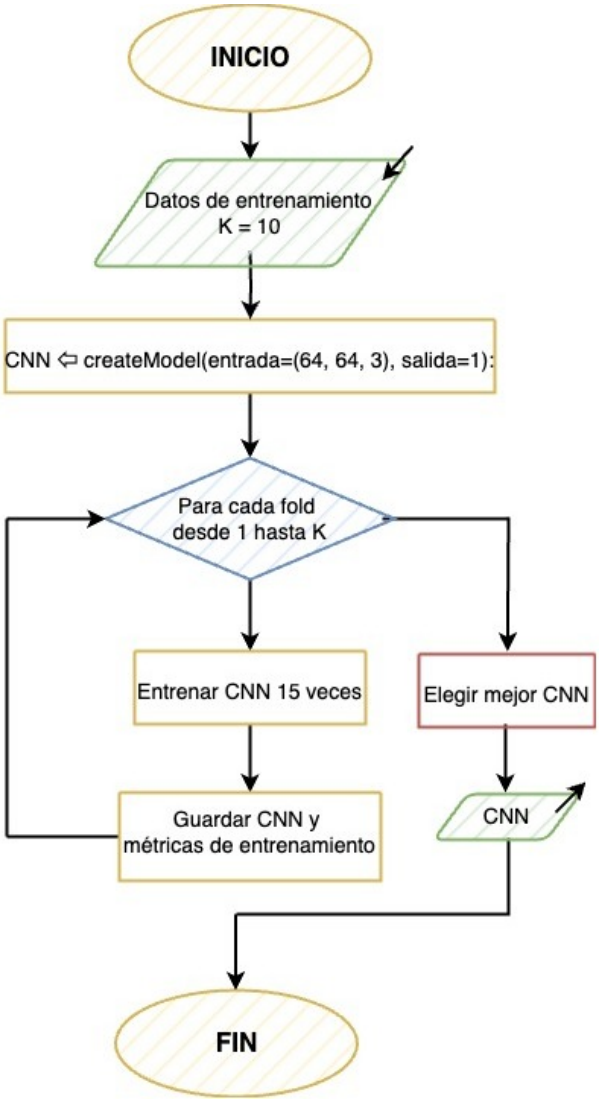


Figura 3.7: Diagrama de flujo de la validación cruzada con enfoque persistente.

`def loadData(path):` Este módulo carga el conjunto de datos a través de una ruta especificada. Posteriormente aplica aumento de datos y los devuelve para ser usados en el entrenamiento y pruebas.
`def StatifiedKFold(splits=10):` Este módulo crea un objeto *KFold* para implementar la validación cruzada estratificada con 10 pliegues.
`def createModel(input_shape=(64, 64, 3), input_shape=1):` Este módulo crea un modelo CNN para entrenar durante todo el proceso con una entrada y salida establecida.
`def crossValidation(data, model, KFold):` Aplica el método de validación cruzada a un modelo utilizando el conjunto de datos y la instancia de *Kfold* especificada. Además, guarda los modelos entrenados en formato h5 utilizados en cada uno de los pliegues.
`def predictByBatch(model, pathBatch, classValue, classes):` Esta función toma un modelo y evalúa un conjunto de datos utilizando la ruta especificada, teniendo en cuenta los valores de las clases proporcionadas como parámetros.
`def testModels(test_set):` Esta función carga todos los modelos guardados en formato h5 presentes en la carpeta raíz, los evalúa con el lote de pruebas pasado como parámetro y muestra diversas métricas, incluyendo el reporte de clasificación y la matriz de confusión.

Tabla 3.9: Módulos utilizados en la validación cruzada.

Capítulo 4

Desarrollo del prototipo de aplicación móvil

El problema de cargar un modelo de redes neuronales convolucionales previamente entrenado en una aplicación móvil presenta desafíos tanto técnicos como de integración. Se ha utilizado React Native, una plataforma popular y muy bien documentada de desarrollo de aplicaciones móviles, con ésta es posible crear aplicaciones que funcionen tanto en dispositivos Android como en dispositivos iOS. El objetivo específico es garantizar que la aplicación, ejecutándose sobre el dispositivo móvil, pueda aprovechar el poder del modelo CNN para la tarea de clasificación. Este proceso implica no solo la correcta implementación del modelo en el entorno de React Native, sino también la optimización para el rendimiento y la gestión de los recursos disponibles en los dispositivos móviles, que suelen tener limitaciones de memoria y potencia de procesamiento.

En el presente capítulo se detallan los aspectos importantes para el desarrollo del prototipo de una aplicación móvil. En la Sección 4.1 se abordan las especificaciones tanto del hardware como del software utilizados. La Sección 4.2 explora el desarrollo del prototipo de la aplicación móvil utilizando *React Native*.

4.1. Especificaciones de hardware y software

El desarrollo del prototipo de la aplicación móvil se llevó a cabo en una computadora *MacBook Air M1 2020*. Las características de la computadora se detallan en la Tabla 3.1.

Las dependencias utilizadas para el desarrollo del prototipo de aplicación móvil fueron las siguientes:

- *React Native* versión **0.72.4**
- *React* versión **18.2.0**
- *React FS* versión **2.20.0**
- *React Native Async Storage* versión **1.18.2**
- *React Native Heroicons* versión **3.2.0**
- *JPEG-js* versión **0.4.4**
- *Expo* versión **49.0.8**
- *Expo Camera* versión **13.4.2**
- *Expo GL* versión **13.0.1**
- *Expo GL CPP* versión **11.4.0**
- *Expo Image Manipulator* versión **11.3.0**
- *Expo Image Picker* versión **14.3.2**
- *TensorFlow TFJS* versión **4.10.0**
- *Tensorflow TFJS React Native* versión **0.8.0**
- *TypeScript* versión **5.1.3**

4.2. Arquitectura del prototipo de aplicación móvil

La clasificación de tilapias por sexo tradicionalmente requiere la presencia de un experto en el campo para llevar a cabo la predicción del género de los peces de manera manual, un proceso que conlleva costos y consume tiempo. Sin embargo, la incorporación del modelo ML usando CNN en dispositivos móviles presenta una serie de ventajas que puede transformar por completo la complejidad de esta tarea.

Una aplicación basada en CNN en un dispositivo móvil puede facilitar la clasificación del sexo de las tilapias, haciéndola accesible para una variedad más amplia de usuarios, no sólo para los expertos en el campo. Además, su portabilidad permite el uso de la aplicación móvil en cualquier lugar. A esto se suma el hecho de que puede ejecutarse en una gran variedad de dispositivos con sistema operativo *Android*. La aplicación móvil reduce significativamente el tiempo de espera, ya que proporciona resultados en cuestión de segundos a partir de una imagen capturada por la cámara. Esto no sólo optimiza la eficiencia y la velocidad de todo el proceso, sino que también contribuye a la reducción de costos al eliminar la necesidad de aplicar elementos como metileno azul en las papilas genitales de las tilapias, un procedimiento que puede ser costoso y requiere recursos adicionales.

Por lo tanto, se optó por desarrollar una aplicación móvil que aprovecha tres tecnologías clave: *Keras* para el entrenamiento del modelo de CNN, *Tensorflow.js* para la conversión y ejecución del modelo en dispositivos móviles, y *React Native* para la creación de la interfaz de usuario. La elección de *React Native* se hizo tras considerar otras opciones como *Kotlin* y *Swift*. En primer lugar, la implementación de *TensorFlow* con *Kotlin* a través de *TensorFlow Lite* aún está en sus primeras etapas y no incluye la mayoría de las funcionalidades disponibles en *TensorFlow* para *Python* o *TensorFlow.js* [Matt Moore, 2024]. Por otro lado, *Swift* es un lenguaje exclusivo de dispositivos iOS, lo que limita la gama de dispositivos móviles en los que se podría utilizar la aplicación [Apple, 2024]. En cambio, *React Native* ofrece una solución multiplataforma, permitiendo la construcción de aplicaciones tanto para *Android* como para *iOS* con un único código fuente [Native, 2023a]. Además, al estar

basado en *JavaScript*, se puede aprovechar al máximo todas las funcionalidades de TensorFlow a través de TensorFlow.js [TensorFlow, 2023b].

Este proyecto se dividió en cinco fases como se observa en la Figura 4.1:

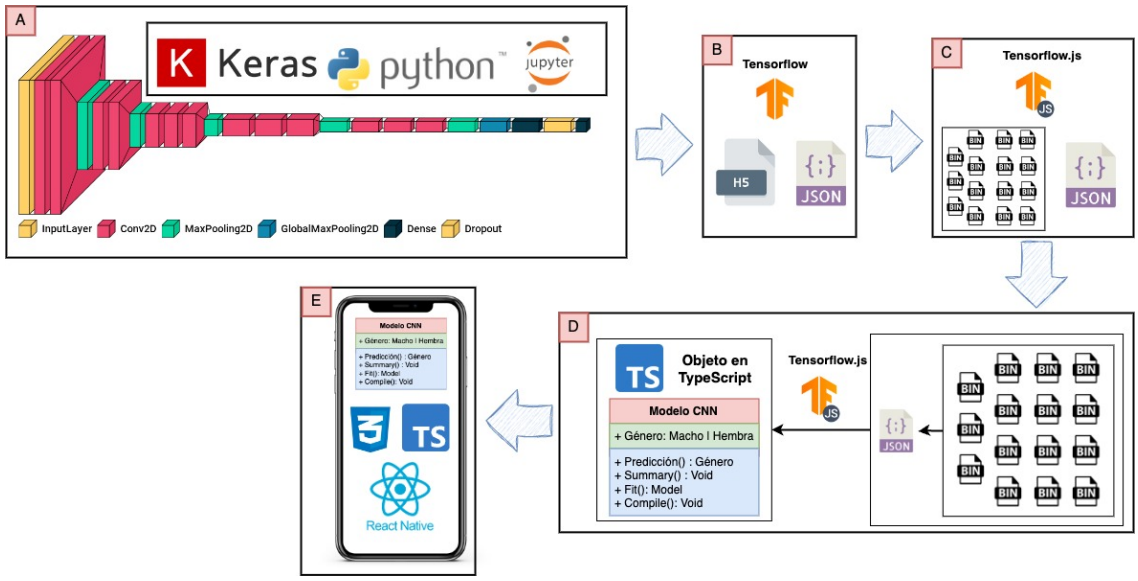


Figura 4.1: Fases para la creación del prototipo: entrenar, exportar y cargar modelos de ML de Keras a un dispositivo móvil.

a) **Modelo entrenado en Keras.** Inicialmente, se realizaron varios experimentos usando CNN en un entorno de desarrollo de aprendizaje profundo utilizando *Keras* y el lenguaje de programación *Python* a través de una libreta de *Jupyter*.

b) **Exportación del modelo.** Una vez realizados los experimentos, se seleccionó el mejor modelo CNN y se exportó por medio de la API de *Keras* que guarda los siguientes componentes:

   - La arquitectura o configuración, que especifica qué capas contiene el modelo y cómo están conectadas.
   - Los valores de peso del modelo.
   - El optimizador y su estado.

La API de Keras guarda todos estos elementos en 2 archivos importantes:

- Un archivo de configuración en formato *JSON* donde se obtiene una representación del modelo en forma de texto, que incluye información sobre su arquitectura, capas, configuración y parámetros.
- Un archivo de estado en el formato Hierarchical Data Format versión 5 (*H5*), que incluye los pesos, representando los valores específicos ajustados por el modelo durante el proceso de entrenamiento [TensorFlow, 2024b].

Ambos archivos se guardan usando el Código 4.1.

```
from keras.applications import Functional

def export_model(model: Functional, name: str) -> None:
    model_json = model.to_json()
    with open(name + ".json", "w") as json_file:
        json_file.write(model_json)
    model.save_weights(name + ".h5")
    print("Saved model to disk")
```

Código 4.1: Forma de guardar, serializar y exportar un modelo de Keras.

c) **Conversión de los archivos con Tensorflow.js.** Para poder utilizar el modelo entrenado en Keras en marcos de trabajo basados en *JavaScript* o *TypeScript* como *React Native*, se requiere transformar el archivo *model.json* y *model.h5* generados en la fase anterior en los siguientes archivos:

- Un archivo de configuración en formato *JSON* que contiene la topología del modelo. La principal diferencia entre el JSON generado por la API de Keras es que ese archivo incluye una representación completa del modelo en forma de texto, incluyendo detalles sobre su arquitectura, capas. hiperparámetros y configuración, mientras que el archivo JSON resultante de la conversión con TensorFlow.js se enfoca en la topología del modelo, describiendo la estructura y la secuencia de capas.
- Uno o más archivos binarios (la cantidad depende de la arquitectura del modelo) con formato *bin* que incluye el conjunto de pesos de las capas.

La conversión mencionada se logra mediante el Código 4.2.

```
from tensorflow.keras.models import model_from_json, Functional
import tensorflowjs as tfjs

def loadModel(nameModel: str) -> Functional:
    json_file = open(nameModel + ".json", 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    cnn = model_from_json(loaded_model_json)
    cnn.load_weights(nameModel + ".h5")
    print("Loaded model from disk")
    return cnn

model = loadModel("model")
tfjs.converters.save_keras_model(model, "carpeta_destino")
```

Código 4.2: Módulo que permite cargar el modelo exportado en la Fase 2 con los archivos *model.json* y *model.h5* que genera un archivo *model.json* y uno o más archivos *bin* dependiendo la arquitectura del modelo.

d) **Conversión del modelo a un objeto de *TypeScript* para ser usado por *React Native*.**

Para integrar el modelo transformado en la fase anterior en un proyecto de *React Native*, se utilizó la API de TensorFlow.js diseñada específicamente para esta tecnología, conocida como *tfjs-react-native API* con la finalidad de poder cargar el modelo y transformarlo a un objeto de *TypeScript* para poder acceder a sus métodos y atributos justo como se puede realizar desde *Python*. Esta API ofrece el módulo *BundleResourceIO*, el cual facilita la carga de modelos y recursos de TensorFlow.js directamente desde el sistema de archivos local de la aplicación móvil [TensorFlow, 2023c]. Esto significa que se pueden incluir modelos en una aplicación móvil y utilizarlos sin necesidad de realizar solicitudes a través de la red para acceder a un modelo en un servidor, lo cual es beneficioso ya que evita costos asociados con la transferencia de datos a través de la red móvil.

En el Código 4.3, se puede observar la importación inicial de las bibliotecas

necesarias, seguida de la carga del archivo *JSON* y los 15 archivos de pesos del modelo en formato *bin*, los cuales fueron generados en la fase anterior. Posteriormente, el modelo se carga utilizando *tf.loadLayersModel* y la funcionalidad proporcionada por *BundleResourceIO*. Finalmente, se devuelve una instancia del modelo cargado para su posterior uso en la aplicación.

```
import * as tf from "@tensorflow/tfjs";
import {bundleResourceIO} from "@tensorflow/tfjs-react-native";

const modelJson = require("../vgg16Model/model.json");

const modelWeights = [
  require("../vgg16Model/group1-shard1of15.bin"),
  require("../vgg16Model/group1-shard2of15.bin"),
  .
  . // Rest of weights
  .
  require("../vgg16Model/group1-shard14of15.bin"),
  require("../vgg16Model/group1-shard15of15.bin"),
];

export const loadModel = async () => {
  await tf.ready();
  const model = await tf.loadLayersModel(
    bundleResourceIO(modelJson, modelWeights),
  );
  console.log(model.summary());
  return model;
};
```

Código 4.3: Módulo que permite cargar el modelo exportado en el paso anterior en una aplicación de React Native.

Una vez cargado el modelo, fue posible realizar predicciones con el modelo a partir de imágenes capturadas por la cámara del dispositivo móvil, esto fue gracias al módulo *handlePredictResult* que define una función asincrónica que toma la URI de una foto y un modelo *TensorFlow.js* como entrada y devuelve un objeto *PredictionResult*. Esta función realiza varias operaciones, como

cambiar el tamaño de la foto, decodificar la imagen, normalizar y realizar una predicción utilizando el modelo proporcionado. La función devuelve el género predicho y el resultado de la predicción como se muestra en el Código 4.4.

```
import * as tf from "@tensorflow/tfjs";
import { decodeJpeg } from "@tensorflow/tfjs-react-native";
import * as FileSystem from "expo-file-system";
import * as ImageManipulator from "expo-image-manipulator";

export interface PredictionResult {
  gender: string;
  result: number;
}

const resizePhoto = async (uri: string, size: number[]) => {
  const actions = [{ resize: { width: size[0], height: size[1] } }];
  const saveOptions = {
    base64: true,
    format: ImageManipulator.SaveFormat.JPEG,
  };
  return await ImageManipulator.manipulateAsync(uri, actions, saveOptions);
};

export async function handlePredictPhoto(
  photoUri: string,
  model: tf.LayersModel | undefined,
): Promise<PredictionResult> {
  const { uri } = await resizePhoto(photoUri ?? "", [300, 300]);

  const imgB64 = await FileSystem.readAsStringAsync(uri, {
    encoding: FileSystem.EncodingType.Base64,
  });
  const imgBuffer = tf.util.encodeString(imgB64, "base64").buffer;
  const raw = new Uint8Array(imgBuffer);
  const imagesTensor = decodeJpeg(raw, 3);

  const processedTensor = tf.image.resizeBilinear(
    imagesTensor,
    [64, 64],
    false,
  );

  const expandedTensor = tf.expandDims(processedTensor, 0);
  const normalizedTensor = tf.div(expandedTensor, tf.scalar(255));

  const prediction = (await model?.predict([
    normalizedTensor,
  ])) as tf.Tensor<tf.Rank>;

```

```
46    const result = await prediction.data();
47    console.log("Predict " + result.toString());
48
49    tf.dispose([imagesTensor, processedTensor, expandedTensor, prediction]);
50
51    const gender = result[0] < 0.5 ? "Hembra" : "Macho";
52
53    return { gender, result: result[0] };
54  }
```

Código 4.4: Módulo que permite realizar una predición a partir de una imagen obtenida por la cámara.

Es importante destacar que originalmente, el proceso de predicción requería aproximadamente de 30 a 40 segundos en promedio debido a la calidad de la imagen de entrada. Sin embargo, gracias al módulo *ImageManipulator* de Expo, que permite realizar manipulaciones y ajustes a imágenes de manera eficiente [Expo, 2023], se redujo significativamente la cantidad de píxeles a tan solo 300×300. Este módulo facilita la manipulación de la imagen, lo que resultó en una reducción significativa del tiempo de procesamiento a solo 1 segundo en promedio, dicho módulo se implementa dentro de la función *resizePhoto* como se observa en el Código 4.4.

En la Tabla 4.1 se presenta una comparativa de los tiempos de predicción con y sin la implementación del módulo de redimensión. Se destaca una reducción significativa en el tiempo de ejecución al hacer uso de dicho módulo. Esto se pudo comprobar mediante una serie de pruebas realizadas con dos dispositivos: un *Huawei Y9 2019*, basado en el sistema operativo *Android* y con una cámara de 13 megapíxeles y un *iPhone SE 2020*, basado en el sistema operativo *iOS* y con una resolución de cámara de 12 megapíxeles.

El análisis de los datos proporcionados revela una reducción significativa en los tiempos de ejecución al implementar la optimización de redimensión de imágenes en ambos dispositivos. A partir de estos datos, es posible calcular el porcentaje de disminución para ambos casos.

$$\left(\frac{31.8508 - 1.0502}{31.8508}\right) \times 100\,\% = 96.7027 \tag{4.1}$$

	Tiempo sin redimensión (s)	Tiempo con redimensión (s)
iPhone SE 2020	26.967	1.046
	26.651	0.973
	25.952	0.884
	30.421	0.959
	42.427	0.925
	25.343	1.899
	26.670	0.963
	39.047	0.997
	34.434	0.972
	40.596	0.884
Huawei Y9 2019	47.131	1.234
	50.428	1.412
	43.673	1.544
	32.181	1.514
	37.676	1.239
	47.601	1.562
	29.124	1.209
	49.484	1.479
	45.681	1.124
	45.041	1.488

Huawei Y9 2019	Tiempo sin redimensión (s)	Tiempo con redimensión (s)
Promedio	42.802	1.380
iPhone SE 2020	Tiempo sin redimensión (s)	Tiempo con redimensión (s)
Promedio	31.8508	1.0502

Tabla 4.1: La columna *Tiempo sin redimensión* muestra los tiempos de ejecución originales sin ninguna optimización, mientras que la columna *Tiempo con redimensión* representa los tiempos después de aplicar la optimización de redimensión de imágenes. Estos resultados fueron obtenidos utilizando un dispositivo *Huawei Y9 2019* y otro dispositivo *iPhone SE 2020*.

$$\left(\frac{42.802 - 1.380}{42.802}\right) \times 100\,\% = 96.7758 \tag{4.2}$$

En el caso del *iPhone SE 2020*, se observa una reducción del 96.7 % en el tiempo de ejecución como se muestra en la Ecuación 4.1, pasando de un promedio de 31.8508 segundos a tan solo 1.0502 segundos. Por otro lado, en el *Huawei Y9 2019*, la reducción es del 96.8 % como se muestra en la Ecuación 4.2, con un tiempo promedio de ejecución disminuido de 42.802 segundos a 1.380 segundos. Esta similitud podría sugerir que la optimización de redimensión de imágenes tiene un impacto consistente en diferentes dispositivos y sistemas operativos, independientemente de las diferencias en hardware o software. Además, la cercanía de estos porcentajes podría indicar una correlación entre la implementación de la optimización y la disminución en los tiempos de ejecución.

Estos resultados demuestran claramente la eficacia de la optimización implementada, no sólo en dispositivos *iOS*, sino también en dispositivos *Android*. Esta mejora en el rendimiento se logra sin comprometer la calidad de las predicciones, lo que resulta en una mejor utilidad y aplicabilidad de la solución propuesta en ambos sistemas operativos.

e) **Creación de la interfaz de usuario**

La interfaz de usuario se ha desarrollado utilizando *React Native*, debido a su capacidad para diseñar aplicaciones con *JavaScript* y *CSS*, además de su compatibilidad con las funcionalidades de *TensorFlow* a través de *TensorFlow.js*. *React Native*, se basa en el concepto de componentes, donde cada elemento en pantalla acepta propiedades de estilo. Esto significa que se puede diseñar la aplicación utilizando *JavaScript* y aplicar estilos de manera coherente, siguiendo una convención de nomenclatura en *camelCase*, como marginTop en lugar de margin-Top, similar al *CSS* tradicional usado en web [Native, 2023b].

Con el objetivo de establecer un flujo de trabajo intuitivo para los usuarios finales, se ha propuesto el diseño de un diagrama que ilustra el flujo de pantallas, tal como se presenta en la Figura 4.2.

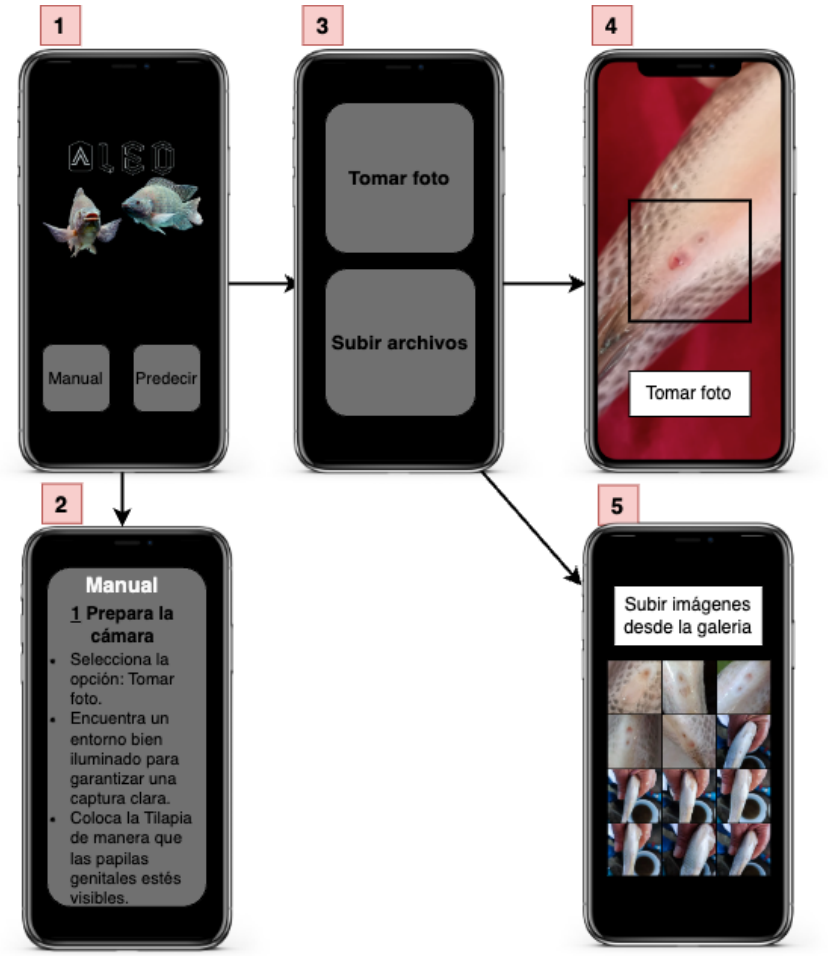


Figura 4.2: Diseño de las pantallas del prototipo de aplicación móvil.

En dicho diagrama podemos observar 5 pantallas enumeradas que se explican a continuación.

1) **Pantalla de inicio**: los usuarios encontrarán el logotipo del prototipo acompañado de dos botones. El primero, destinado a acceder al manual, y el segundo, que proporciona la opción de realizar predicciones.

2) **Manual de usuario**: se presenta el manual detallado del prototipo. Su propósito es brindar a los usuarios la información necesaria para utilizar las herramientas de predicción. El manual abarca instrucciones tanto para capturar imágenes mediante la cámara como para cargar imágenes desde

la galería.

3) **Elección del tipo de predicción**: los usuarios se encuentran con las dos opciones de predicción disponibles: mediante la cámara o seleccionando imágenes desde la galería. Aquí, los usuarios pueden optar por cualquiera de estas dos herramientas según sus preferencias.

4) **Predicción tomando una imágen por cámara**: se habilita la funcionalidad para realizar predicciones utilizando imágenes capturadas mediante la cámara. En esta pantalla, los usuarios encontrarán un recuadro para enfocar la papila genital de la tilapia, un botón para capturar la imagen y la opción de utilizar la función de zoom de la cámara. Una vez que la cámara del dispositivo móvil ha capturado la imagen, se despliega la interfaz que se muestra en la Figura 4.3, la cual consta de cuatro elementos principales: la propia imagen capturada y recortada para eliminar contenido innecesario, un botón con la inscripción "Predecir"que permite obtener el sexo del pez basándose en la imagen capturada, otro botón con la etiqueta "Descartar"que posibilita tomar una nueva imagen en caso de captura defectuosa o para realizar una nueva predicción, y finalmente, una etiqueta de clase que indicará el sexo del pez.

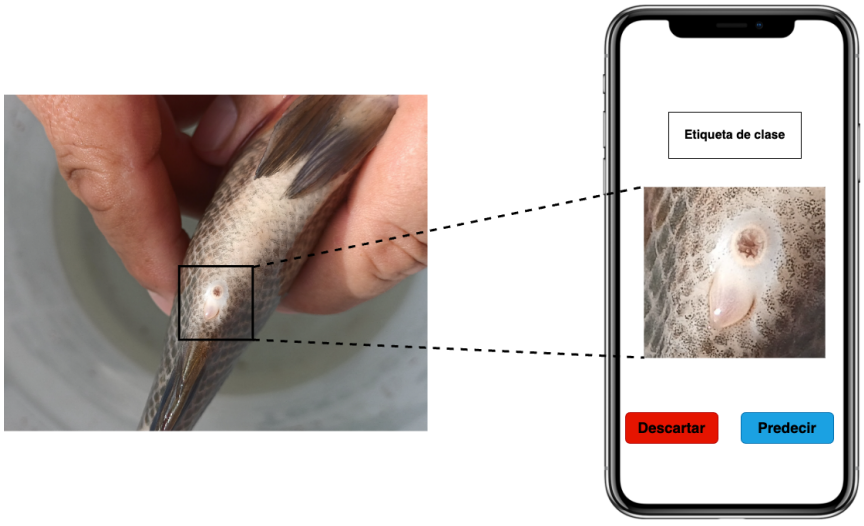


Figura 4.3: Diseño de la interfaz para predecir el sexo de la tilapia en base a una imagen capturada con la cámara.

5) **Predicción subiendo imágenes desde la galería**: en esta sección, los usuarios tienen la opción de utilizar la funcionalidad de predicción con imágenes seleccionadas desde la galería, gracias a esto se pueden elegir varias imágenes a la vez como se aprecia en la Figura 4.4, además de poder recortar el área de interés de forma manual.

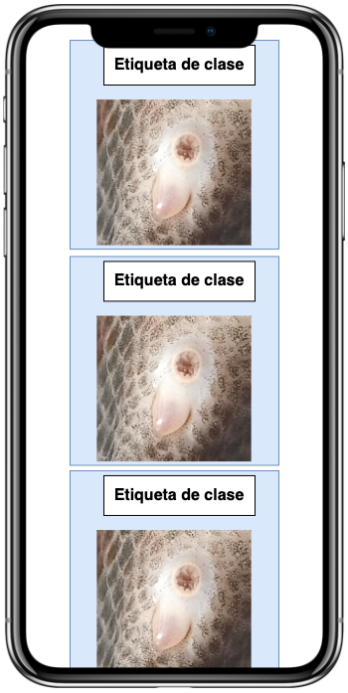


Figura 4.4: Diseño de la interfaz para predecir múltiples imágenes cargadas desde la galería.

Finalmente, el flujo de predicción se ha definido como se muestra en la Figura 4.5. Este flujo comienza con la introducción de una imagen de entrada, ya sea capturada mediante la cámara del dispositivo o seleccionada desde la galería de imágenes. A continuación, la imagen pasa por un proceso de recorte, que puede ser realizado manualmente o de manera automática, con el objetivo de prepararla para su posterior procesamiento. Posteriormente, la imagen es redimensionada y separada en sus distintos canales de color. Luego, se transforma en un tensor y se normalizan sus valores para finalmente ser proporcionada como entrada al modelo CNN. Como re-

sultado de este proceso, se obtiene la predicción del sexo del pez basada en la imagen capturada, utilizando el resultado discreto del modelo CNN.

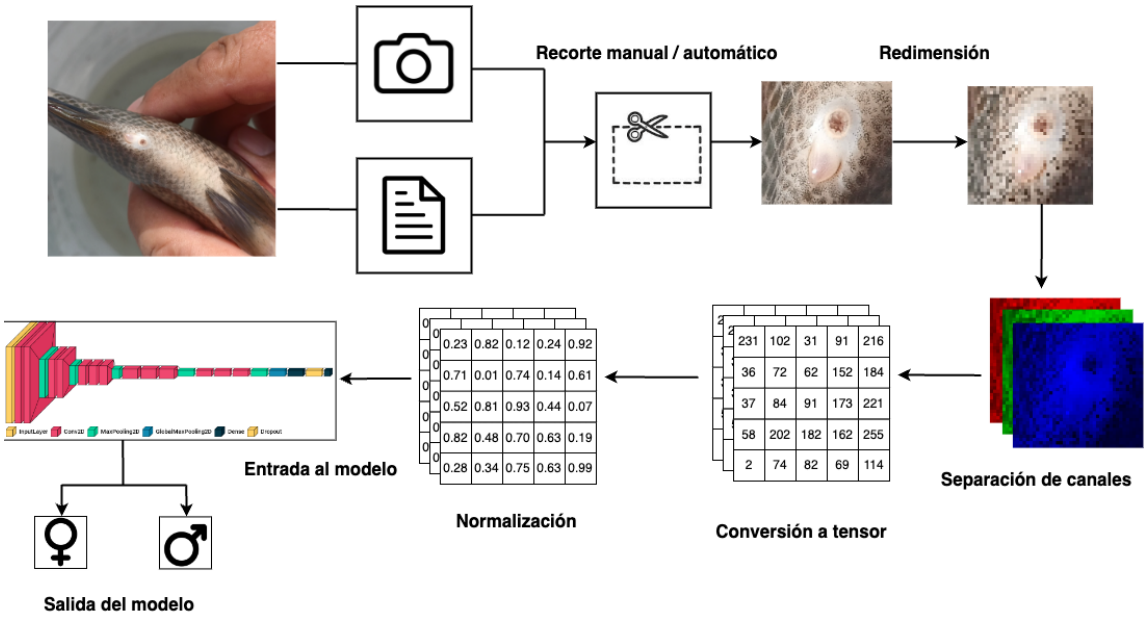


Figura 4.5: Flujo de predicción usando el modelo CNN a partir de una imagen de entrada.

Capítulo 5

Resultados

Este capítulo presenta los resultados derivados de las experimentaciones detalladas en el capítulo anterior. En la Sección 5.1, se muestra la arquitectura de la CNN seleccionada haciendo una comparación entre los resultados obtenidos al utilizarlas arquitecturas *VGG-16* y *LeNet-5*. Posteriormente, en la Sección 5.2, se muestran las configuraciones óptimas de la red, las cuales se determinaron tras una búsqueda exhaustiva de los hiperparámetros que maximizan la capacidad de generalización del modelo. En la Sección 5.3, se presenta la elección final del mejor modelo, la cual se realizó tras un análisis de las métricas de clasificación obtenidas de una experimentación usando validación cruzada. El modelo seleccionado se carga en la aplicación móvil y su uso se detalla en la Sección 5.4. En esta sección, se describe cómo se utiliza el modelo CNN seleccionado en un prototipo de aplicación móvil desarrollado con *React Native*, así también se presenta la interfaz de usuario. En la Sección 5.5, se detallan los pasos realizados para desplegar la aplicación móvil en la tienda de Google Play. Finalmente, en la Sección 5.6, se muestran los resultados de las pruebas de campo realizadas con el prototipo de aplicación móvil en cultivos de tilapias.

5.1. Elección de la arquitectura CNN

Tras realizar un conjunto de experimentos para seleccionar la arquitectura CNN más adecuada en términos del rendimiento de clasificación, se compararon las arquitecturas *VGG-16* y *LeNet-5*. Se entrenaron 5 modelos de cada una de estas arquitecturas durante 100 épocas para realizar un análisis de su rendimiento.

Los resultados obtenidos, que se observan en la Tabla 5.1, muestran que la arquitectura *VGG-16* superó a *LeNet-5* en términos de rendimiento de clasificación. *VGG-16* demostró ser más efectiva, logrando un promedio de 0.9525 de exactitud y 0.1465 de pérdida, mientras que *LeNet-5* obtuvo un promedio de 0.89166 de exactitud y 0.24298 de pérdida. Este resultado sugiere que la complejidad y profundidad de la arquitectura *VGG-16* pueden haber sido beneficiosas para la tarea de clasificación en el conjunto de datos.

Arquitectura	#	**Entrenamiento**		**Prueba**	
		Pérdida	**Exactitud**	**Pérdida**	**Exactitud**
LeNet-5	1	0.1966	0.9132	0.2571	0.8750
	2	0.1904	0.9222	0.2153	0.9083
	3	0.1890	0.9193	0.2453	0.8958
	4	0.1992	0.9148	0.2682	0.8792
	5	0.1683	0.9264	0.2290	0.9000
Promedio		**0.1887**	**0.91918**	**0.24298**	**0.89166**
VGG-16	6	0.0535	0.9788	0.1386	0.9625
	7	0.0457	0.9823	0.1046	0.9625
	8	0.0544	0.9785	0.1607	0.9542
	9	0.0564	0.9781	0.1725	0.9500
	10	0.0653	0.9762	0.1561	0.9333
Promedio		**0.05506**	**0.97878**	**0.1465**	**0.9525**

Tabla 5.1: Métricas de la experimentación con las arquitecturas *LeNet-5* y *VGG-16*. Se muestran los resultados de pérdida (*loss*) y exactitud (*accuracy*) tanto para el conjunto de entrenamiento como para el conjunto de prueba.

Aunado a los resultados anteriores, en la Figura 5.1 se muestra el comportamiento de las métricas mostradas en la Tabla 5.1 a través de las 100 épocas de

entrenamiento. Se muestra gráficamente la diferencia en cuanto a desempeño entre ambas arquitecturas a lo largo del proceso de entrenamiento. Se observa que la arquitectura *VGG-16* alcanzó una convergencia más rápida y obtuvo un rendimiento generalmente superior en comparación con *LeNet-5*. La brecha entre los valores de pérdida obtenidas con los datos de entrenamiento y los correspondientes de prueba también es más estrecha en el caso de *VGG-16*, lo que sugiere una mejor capacidad de generalización. Estos resultados respaldan aún más la superioridad de la arquitectura *VGG-16* en esta tarea de clasificación.

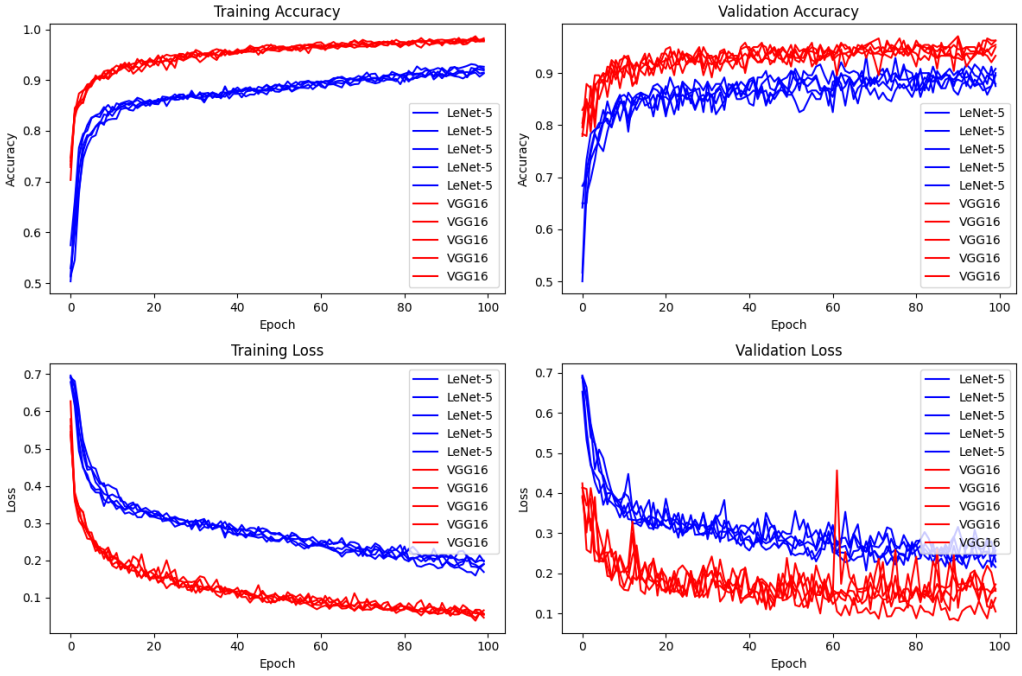


Figura 5.1: Métricas de los 10 modelos con arquitecturas *LeNet-5* y *VGG-16* a través de 100 épocas.

5.2. Ajuste de hiperparámetros

Una vez identificada la arquitectura *VGG-16* como la más idónea para abordar el problema de clasificación en cuestión, se dio paso a la siguiente etapa: el ajuste

de hiperparámetros. En esta fase, se llevó a cabo una búsqueda exhaustiva en malla, explorando diversas combinaciones.

Este proceso dio como resultado la creación de un total de 36 modelos, todos basados en la arquitectura *VGG-16*. De estos 36 modelos, se presentan los 5 mejores en la Tabla 5.2 y el resto de configuraciones se muestra en el Apéndice B.

Como resultado de esta búsqueda, se identificaron los mejores conjuntos de hiperparámetros. Estos conjuntos representan combinaciones que generaron los modelos con el mayor rendimiento en términos de exactitud (*Accuracy*) y pérdida (*Loss*). En la Tabla 5.2 se presentan los hiperparámetros de las cinco configuraciones con los mayores valores de exactitud.

Tasa de aprendizaje	**Eliminación aleatoria (Dropout)**	**No. de épocas**	**Unidades**	**exactitud**	**Pérdida**
0.0005	0.3	100	1024	**0.9666**	**0.1277**
0.0005	0.5	100	512	0.9624	0.1183
0.0005	0.3	80	1024	0.9583	0.0878
0.0005	0.4	80	512	0.9624	0.1634
0.0005	0.4	100	1024	0.9541	0.1135

Tabla 5.2: Resultados de las 5 mejores configuraciones en la búsqueda y ajuste de hiperparámetros.

Además de los datos de la Tabla 5.2, mediante la plataforma *Weight y Biases* se obtuvieron más detalles sobre esta experimentación. En la Figura 5.2 destaca que la *Tasa de Aprendizaje* (del inglés *Learning Rate*) es el hiperparámetro con mayor correlación e importancia del modelo en relación con la función de pérdida, indicando que pequeñas variaciones en la *Tasa de Aprendizaje* pueden tener un impacto significativo en el rendimiento del modelo. Una *tasa de aprendizaje* adecuada permite que el modelo aprenda a un ritmo óptimo: ni demasiado rápido, lo que podría llevar a un aprendizaje inestable y a una falta de convergencia; ni demasiado lento, lo que podría resultar en un entrenamiento innecesariamente largo o incluso en el estancamiento del modelo en un mínimo local subóptimo. En la gráfica de la Figura 5.3, se observa visualmente que una *tasa de aprendizaje* de 0.0025 resulta en una función

de pérdida más alta, lo que podría generar un sobreajuste del modelo. Por otro lado, una *Tasa de Aprendizaje* de 0.0005 ha demostrado ser el valor más efectivo, ya que las mejores 5 configuraciones lo utilizan.

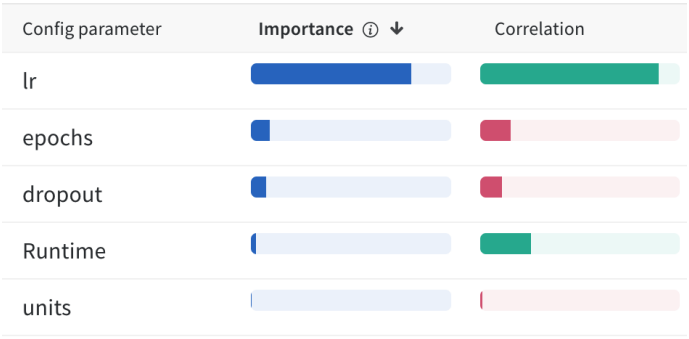


Figura 5.2: Correlación e importancia de los hiperparámetros con la función de pérdida.

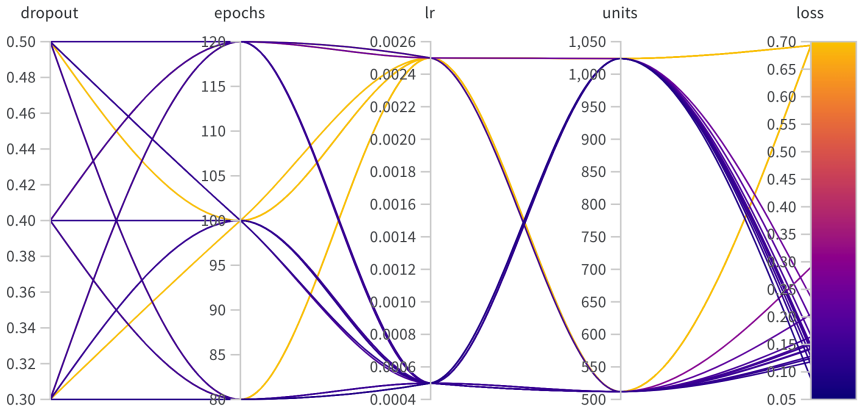


Figura 5.3: Todas las configuraciones de los hiperparámetros teniendo como objetivo la función de pérdida.

Adicionalmente, al analizar la función de pérdida, se realizó un análisis de la importancia de los hiperparámetros y su correlación con la variable objetivo (exactitud) como se muestra en la Figura 5.4. Esta gráfica revela que el número de épocas de entrenamiento y la tasa de aprendizaje son los hiperparámetros con mayor importancia; asimismo la tasa de aprendizaje es el hiperparámetro con mayor correlación con la variable objetivo. La gráfica de la Figura 5.5 presenta todas las configuraciones de la red teniendo como objetivo la exactitud. Estos análisis adicionales refuerzan la importancia de la *Tasa de Aprendizaje* en el rendimiento general del modelo, sugiriendo

que es un hiperparámetro crítico a considerar. Además, la influencia de las épocas de entrenamiento en la exactitud destaca la necesidad de encontrar un equilibrio entre el número de épocas y el resto de hiperparámetros para incrementar el rendimiento.

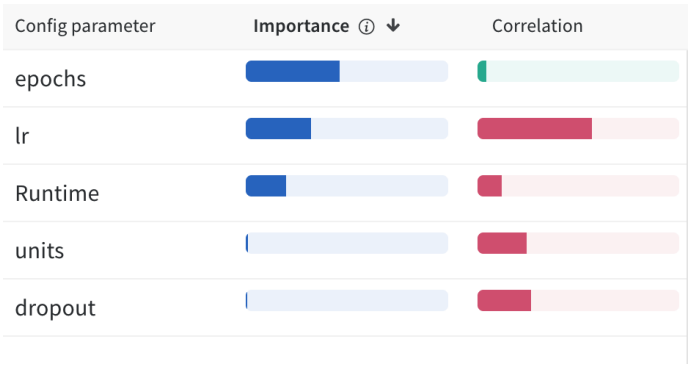


Figura 5.4: Correlación e importancia de los hiperparámetros con la exactitud.

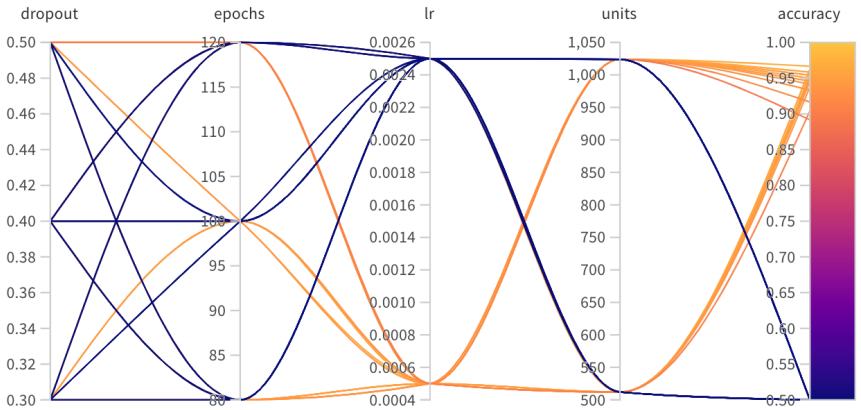


Figura 5.5: Todas las configuraciones de los hiperparámetros teniendo como objetivo la exactitud.

Este análisis de correlación y de importancia de los hiperparámetros nos permite entender mejor cómo cada hiperparámetro afecta el rendimiento del modelo y cómo se pueden ajustar para optimizar la exactitud.

En conclusión, después del análisis de estos resultados se han identificado los conjuntos de parámetros que generaron modelos con el mayor rendimiento de clasificación obtenido en términos de exactitud y pérdida. Entre las cinco mejores configuraciones, el modelo más destacado se obtuvo con una tasa de aprendizaje (*learning*

rate) de 0.0005, un valor de eliminación aleatoria (*dropout*) de 0.3, una duración de entrenamiento de 100 épocas y una capa densa con 1024 unidades. Estos parámetros serán utilizados en la fase siguiente con la finalidad de incrementar el rendimiento de clasificación del modelo.

5.3. Validación cruzada

Una vez obtenidos los hiperparámetros que optimizan la capacidad de generalización de la arquitectura *VGG-16* en el problema de clasificación en cuestión, se procedió a realizar la validación cruzada en la cual se obtuvieron los resultados por cada pliegue (*fold*) mostrados en la Tabla 5.3.

Pliegue	**Exactitud**	**Sensitividad**	**Especificidad**	**Precisión**	**F1-Score**	**MCC**
1	0.9375	0.9083	0.9667	0.9646	0.9356	0.8765
2	0.9292	0.9750	0.8833	0.8931	0.9323	0.8620
3	0.9208	0.9333	0.9083	0.9106	0.9218	0.8419
4	0.9458	0.9417	0.9500	0.9496	0.9456	0.8917
5	0.9417	0.9750	0.9083	0.9141	0.9435	0.8853
6	0.9583	0.9417	0.9750	0.9741	0.9576	0.9172
7	0.9417	0.9583	0.9250	0.9274	0.9426	0.8838
8	0.9667	0.9583	0.9750	0.9746	0.9664	0.9335
9	0.9833	0.9833	0.9833	0.9833	0.9833	0.9666
10	0.9583	0.9833	0.9333	0.9365	0.9593	0.9178
Promedio	**0.9483**	**0.9558**	**0.9408**	**0.9427**	**0.9488**	**0.8976**
Desv. Est.	**0.0176**	**0.0232**	**0.03259**	**0.02959**	**0.01714**	**0.0346**

Tabla 5.3: Métricas de clasificación de cada pliegue.

Los resultados de la validación cruzada muestran un rendimiento consistente del modelo CNN en el problema de clasificación en cuestión. Se observa que en cada uno de los 10 pliegues el modelo logra altos valores de exactitud, sensibilidad, especificidad y F1-score, lo que muestra una buena capacidad para clasificar correctamente

las muestras tanto positivas (machos) y negativas (hembras). Estos resultados son respaldados por el alto valor del Coeficiente de Correlación de Matthews (MCC) que indica una fuerte correlación entre las predicciones del modelo y las clases reales.

Es importante destacar que el modelo presenta una buena capacidad de generalización, ya que las métricas de rendimiento se mantienen consistentemente altas en los diferentes pliegues. Esto indica que el modelo no está sobreajustado a un conjunto de datos específico y tiene la capacidad de generalizar bien a datos no vistos.

En la Figura 5.6 se presenta una representación gráfica del rendimiento del modelo CNN durante el proceso de entrenamiento. Esta Figura ilustra el nivel de exactitud alcanzado en cada uno de los pliegues definidos. Es importante observar que el quinto pliegue registró los valores más bajos mientras que el noveno pliegue destacó con los valores más altos.

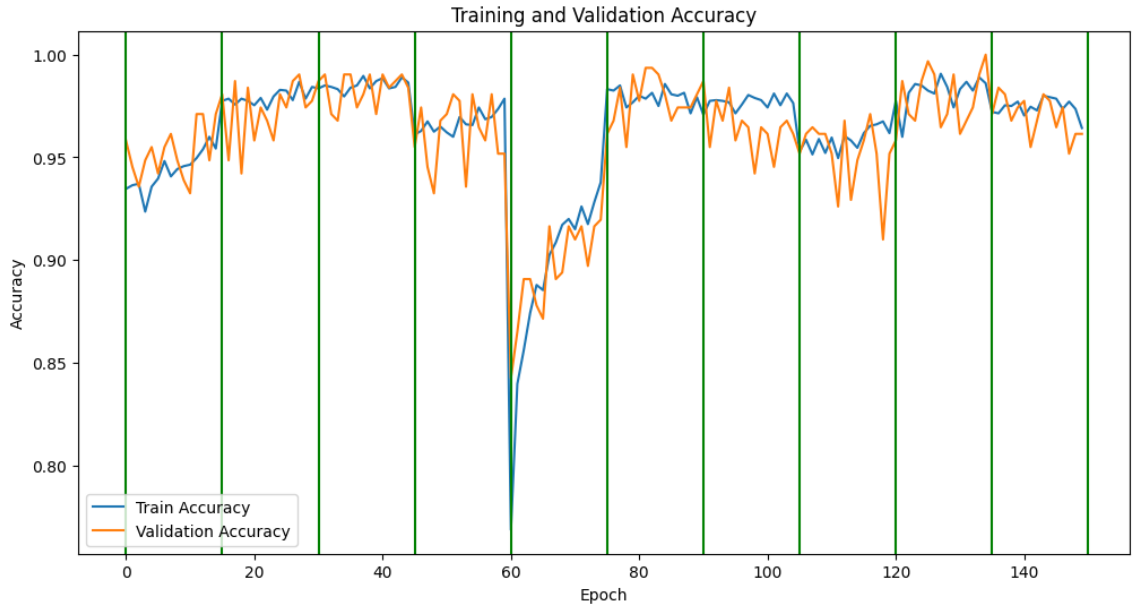


Figura 5.6: Entrenamiento del modelo CNN a través de todos los subconjuntos de datos.

En conclusión, el pliegue 9 presenta un rendimiento superior en comparación con los otros, con una exactitud, sensibilidad, especificidad y F1-score de 0.9833 cada

uno y un MCC de 0.9666 (ver sección 2.5). Además, en la Figura 5.7 se presenta el umbral y los resultados obtenidos por el modelo CNN. Se observa que los datos se concentran en su mayoría en torno a los valores 0 y 1, correspondientes a las dos clases, con pocos datos dispersos o alejados del umbral de decisión. Los resultados muestran solo dos errores por clase, indicando una alta precisión del modelo.

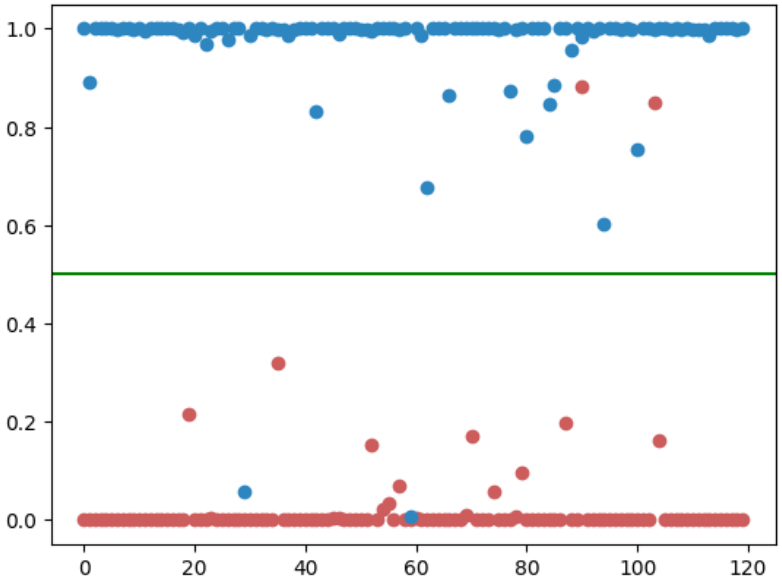


Figura 5.7: Representación gráfica de las respuestas del modelo comparadas con el umbral de clasificación.

Estos resultados indican que el modelo entrenado en el noveno pliegue es altamente confiable y preciso en la identificación de tilapias hembras y machos. Dada la excelencia en el rendimiento del pliegue 9, se espera que el modelo CNN correspondiente sea útil para la aplicación móvil. Debido a su alto nivel de exactitud y fiabilidad, el modelo del noveno pliegue puede desempeñar un papel importante en la clasificación precisa de muestras en tiempo real dentro del entorno de producción.

Se creó una página web que contiene un cuestionario con imágenes del conjunto de prueba para comparar los resultados obtenidos del modelo CNN del pliegue 9 con personas de diversos niveles de experiencia dentro del área, los resultados de este cuestionario se encuentran dentro del Apéndice C. En estos resultados se aprecia que el promedio de exactitud de las personas con mayor experiencia es de 76.90 %

comparado con el 98.33 % obtenido con el modelo del pliegue 9, el cual muestra gran superioridad en comparación de el análisis visual hecho por personas del área que participaron en el cuestionario.

5.4. Prototipo de aplicación móvil

Después de todas las fases anteriores y tras seleccionar el modelo más óptimo para exportar y cargar en un software orientado a dispositivos móviles como *React Native*, se procedió a implementar la interfaz de usuario previamente definida. Así, se generaron las pantallas de inicio, como se muestra en la Figura 5.8.

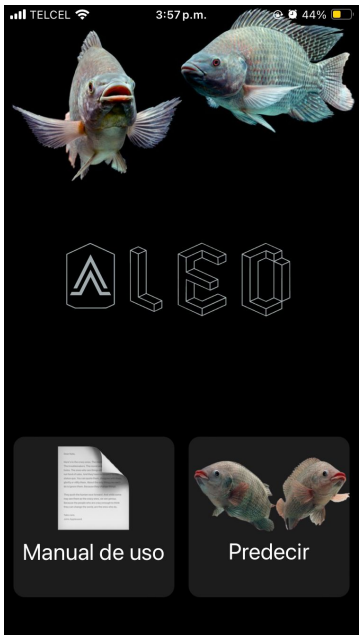


Figura 5.8: Pantalla de inicio realizada con *React Native*.

También, se añadió la pantalla del manual de usuario, dividida en manual de predicción por cámara y manual de predicción por carga de imágenes, tal como se observa en la Figura 5.9; en ella se detalla, de manera intuitiva, los pasos a seguir para lograr clasificar el sexo de una tilapia mediante imágenes.

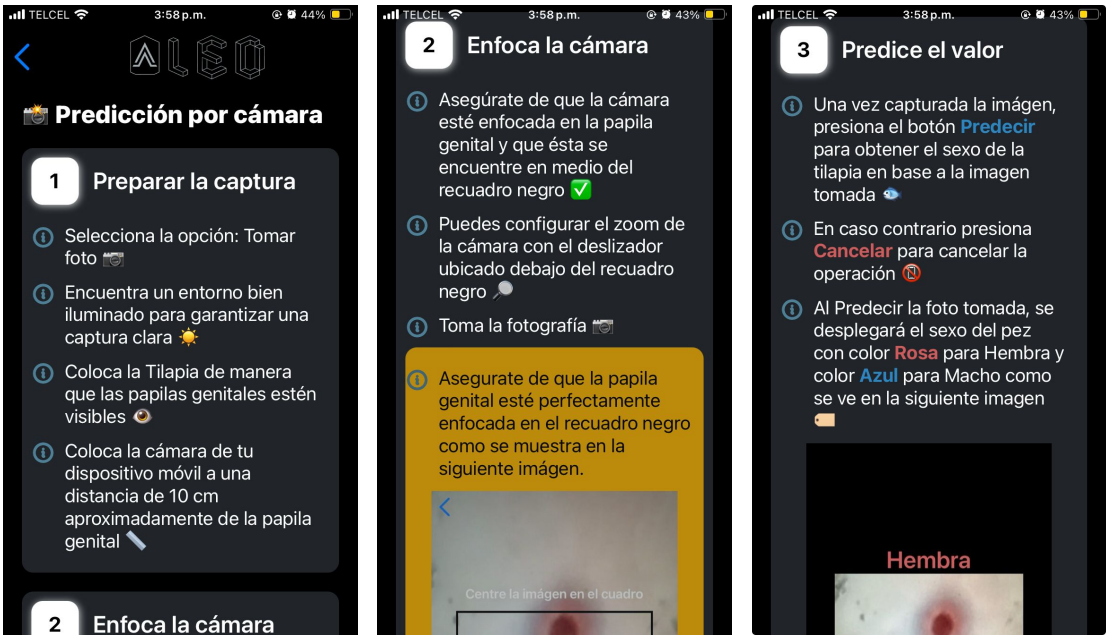
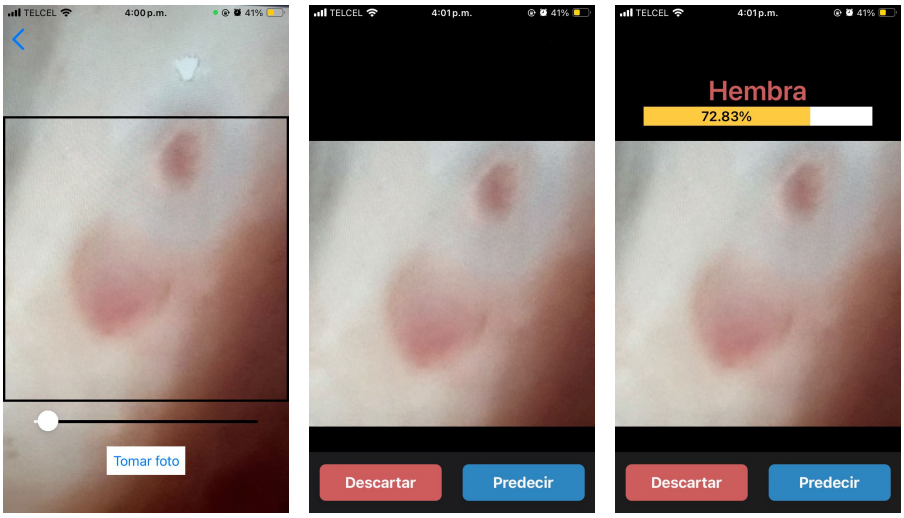


Figura 5.9: Manual de predicción por cámara.

Adicionalmente, se presenta la pantalla de selección de predicción, en la cual se ofrece la opción de cargar imágenes desde la galería o capturar imágenes a través de la cámara, como se muestra en la Figura 5.10.

La función de captura mediante la cámara requiere permisos para su uso. Una vez otorgados, se muestra en pantalla la vista en tiempo real de lo que está siendo capturado a través de la cámara, junto con un cuadro que permite alinear correctamente la papila genital. Además, se proporciona una barra de zoom para facilitar la captura del área de interés tal como se muestra en la Figura 5.11a. Posteriormente, se ofrece un botón para capturar la imagen. Una vez tomada la fotografía, se brinda la opción de realizar la predicción o descartar la imagen para intentarlo de nuevo en caso de algún fallo como se ve en la Figura 5.11b. Una vez realizada la predicción, se muestra la imagen capturada junto con la identificación del sexo de la tilapia, representado en color rosa o azul según corresponda, y una barra de confiabilidad como se ven en al Figura 5.11c.

Figura 5.10: Pantalla para elegir el tipo de predicción.



(a) (b) (c)

Figura 5.11: Toma, recorte y predicción de una imagen obtenida por medio de la cámara de un dispositivo móvil iPhone SE 2020.

La barra de confiabilidad se muestra en la Figura 5.12 y funciona de la siguiente manera:

- **Rango de 0 % a 60 % (Rojo)**: cuando la barra se pinta de color rojo, significa que la confiabilidad de la predicción es relativamente baja. En este caso, el modelo no está seguro de su predicción. Esto puede deberse a varios factores, como la calidad de la imagen, la iluminación, la orientación de la papila genital, entre otros. En estos casos podría ser útil intentar una nueva imagen.

- **Rango de 61 % a 80 % (Amarillo)**: cuando la barra se pinta de amarillo, la confiabilidad de la predicción es moderada, aunque el modelo tiene cierta confianza en su predicción, aún existe una posibilidad significativa de error. En estos casos, si es posible, podría ser útil obtener una nueva imagen para confirmar la predicción.

- **Rango de 81 % a 100 % (Verde)**: cuando la barra se pinta de verde, la confiabilidad de la predicción es alta. Esto significa que el modelo es bastante seguro de su predicción.

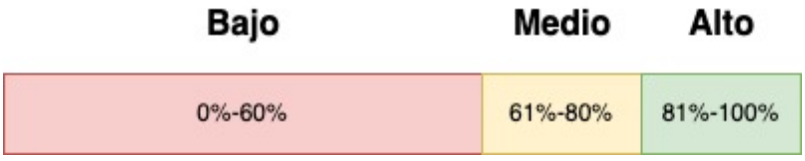
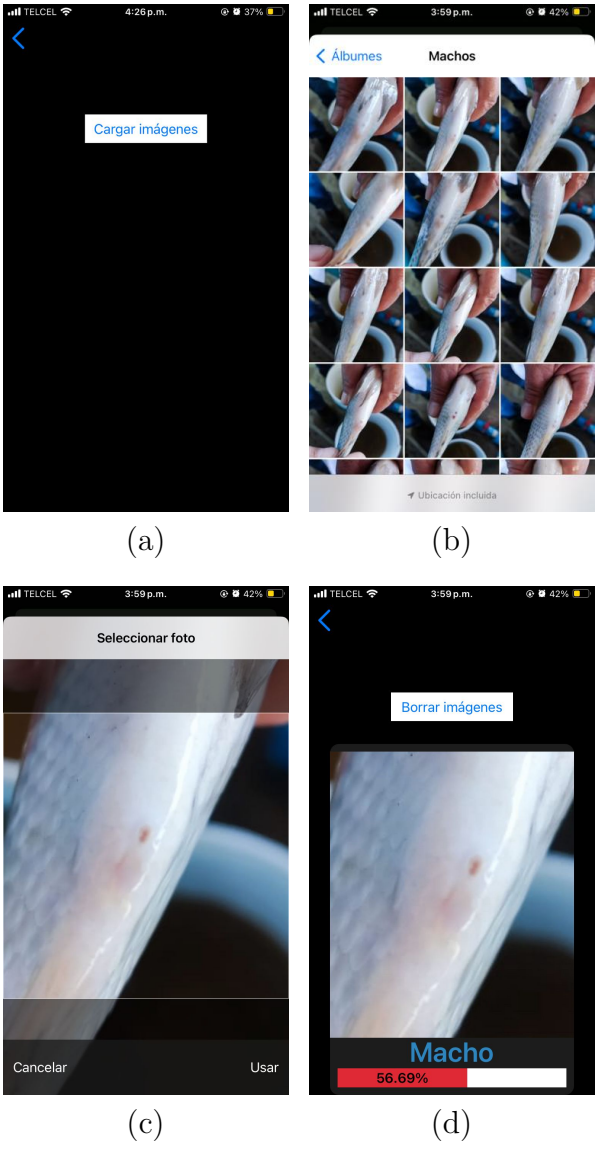


Figura 5.12: Barra de confiabilidad de la predicción.

Por otro lado, la función de predicción mediante imágenes desde la galería ofrece una opción que permite visualizar todas las imágenes disponibles en la galería del dispositivo móvil, tal como se ilustra en las Figuras 5.13a y 5.13b. Esta funcionalidad posibilita la selección de una imagen específica para su recorte manual, proceso que se muestra en la Figura 5.13c.

Una vez realizado el recorte, la galería se vuelve a abrir automáticamente, permitiendo la elección de otra imagen si es necesario. Esto facilita la subida de múltiples imágenes de manera consecutiva. Finalmente, una vez que las imágenes deseadas han sido seleccionadas y recortadas, se presentan junto con sus respectivas predicciones y barras de confiabilidad. Este último elemento, representado en la Figura

5.13d, facilita la rápida evaluación de una gran cantidad de datos, proporcionando una representación visual inmediata de la confiabilidad de cada predicción.



(a) (b)

(c) (d)

Figura 5.13: Manual de predicción por cámara.

5.5. Despliegue del prototipo en Google Play

Tras la finalización del desarrollo del prototipo de aplicación móvil mediante *React Native*, se procedió a llevar a cabo su despliegue en Google Play Store con el objetivo de hacerla accesible para cualquier usuario que desee utilizarla. Este proceso de implementación llevó una serie de pasos detallados a continuación:

5.5.1. Compilación del prototipo

Antes de desplegar el prototipo, fue necesario exportar el proyecto realizado con *React Native*, este fue exportado en formato AAB. AAB (*Android App Bundle*) es el nuevo formato de publicación oficial de *Android*. Este formato permite que se cree una APK (*Android Application Package*) a medida. Google Play utiliza el archivo AAB para generar y servir APKs optimizados para cada configuración del dispositivo, de modo que solo se descargan el código y los recursos necesarios para ejecutar la aplicación en un dispositivo específico. De esta manera, ya no se necesita construir múltiples APKs para dar soporte a diferentes dispositivos como se realizaba antes, y de esta manera los usuarios obtienen descargas más pequeñas y optimizadas [Developers, 2024]. Para realizar esta exportación, se siguieron los siguientes pasos:

a. **Uso de Expo y EAS Build**: Para el proceso de construcción y despliegue se utilizó Expo, una plataforma de código abierto para hacer aplicaciones nativas universales para *Android*, *iOS* y la web con *JavaScript* y *React*. Se creó una cuenta en Expo desde la página oficial [Expo, 2024c] y se utilizó EAS Build, un servicio alojado para construir binarios de aplicaciones para proyectos de Expo y *React Native* [Expo, 2024b]. Para acceder a las funcionalidades de EAS de Expo desde el proyecto de *React Native* se utilizó el comando `npm install -g eas-cli` [Expo, 2024a].

b. **Inicio de sesión con EAS:** Antes de ejecutar los comandos de EAS Build, se utilizó el comando `eas login`. Este comando se utiliza para iniciar sesión con

la cuenta de Expo posteriormente creada. Es un paso fundamental para poder interactuar con los servicios de EAS desde la terminal por medio de la CLI de EAS [Expo, 2024a].

c. **Ejecución de comandos EAS Build:** Desde la raíz del proyecto, se ejecutó el comando `eas build --platform android` esto genera una compilación de la aplicación en formato AAB que se puede utilizar para pruebas internas antes de la publicación final [Expo, 2024a].

5.5.2. Configuración desde Google Play Console

Una vez exportado el proyecto en formato AAB, se continuó con la configuración desde Google Play Console, esta es la plataforma oficial de Google para desarrolladores de aplicaciones *Android*. Una de las funciones más importantes de Google Play Console es la publicación de aplicaciones. Es posible subir aplicaciones a Google Play para que los usuarios puedan descargarlas. Además permite gestionar diferentes versiones de una aplicación y lanzar versiones de prueba o betas.

Para iniciar con la configuración y acceder a todas las funcionalidades de la plataforma, fue necesario la creación de una cuenta en Google Play Console y adquirir la licencia de desarrollador con un costo de 25 USD.

Una vez teniendo acceso a las funcionalidades de Google Play Console, fue posible configurar los siguientes puntos:

1. **Nombre de la aplicación:** es importante elegir el nombre adecuado para el prototipo. Este nombre aparecerá dentro de la consola de *Google Play*. Se seleccionó el nombre "*AlEd*" para el prototipo.

2. **Recursos visuales:** Para publicar el prototipo, se deben cargar los recursos necesarios, como el ícono de la aplicación, gráfico de funciones, imágenes de la

interfaz, entre otros. Estos recursos ayudan a los usuarios a entender rápidamente de qué trata la aplicación y cómo se ve.

3. **Política de privacidad:** Se estableció y se publicó en una página para tener acceso desde cualquier parte. Esto es crucial para informar a los usuarios sobre cómo se manejan sus datos. La política de privacidad se encuentra en el Anexo A.

4. **Acceso a la app:** La app no requiere ningún tipo de credencial de acceso ni ingresar ningún dato. Esto puede hacer que la aplicación sea más accesible para los usuarios.

5. **Anuncios:** Se denegaron. Esto puede mejorar la experiencia del usuario al usar la aplicación, ya que no se distraerá con anuncios.

6. **Clasificación:** Se otorgó *Todas las edades*. Esto significa que la aplicación es adecuada para usuarios de todas las edades.

7. **Grupo etario:** Mayores de 18. Aunque la aplicación es adecuada para todas las edades, está dirigida principalmente a usuarios mayores de 18 años.

8. **Funciones financieras:** No tiene ninguna. Esto significa que la aplicación no maneja transacciones financieras, lo que puede ser un factor de decisión importante para algunos usuarios.

5.5.3. Prueba cerrada

Antes de publicar una aplicación en Google Play, se solicita una prueba cerrada. Esta prueba es una herramienta que ayuda a los desarrolladores a probar sus aplicaciones, identificar problemas, obtener comentarios y asegurarse de que todo esté listo antes del lanzamiento [Play, 2024]. Las pruebas permiten verificar la exactitud, el comportamiento funcional y la usabilidad de la aplicación antes de su lanzamiento público, minimizando el impacto de cualquier problema técnico o de la experiencia del usuario.

Las pruebas cerradas, permiten compartir una aplicación con un grupo selecto de usuarios para solucionar problemas y asegurar que la aplicación satisfaga la política de Google Play antes del lanzamiento. Para poder solicitar el acceso a producción y publicar la aplicación, es necesario ejecutar una prueba cerrada. Cuando se solicita el acceso a producción, al menos 20 verificadores deberán haber participado en la prueba cerrada durante al menos los últimos 14 días de forma continua [Play, 2024].

Para la realización de la prueba cerrada, se solicitó la colaboración de estudiantes de la Universidad Tecnológica de la Mixteca, así como de profesores y personal de la Universidad del Mar. Se solicitó a los participantes que proporcionarán el correo electrónico vinculado a su cuenta de Google Play para incluirlos en la lista de participantes. Tras añadir a todos los usuarios, se les proporcionó un enlace para descargar la aplicación e iniciar la fase de prueba cerrada. Adicionalmente, se les proporcionó un enlace a un sitio web donde se mostraban 10 imágenes por sexo de las tilapias para que pudieran realizar predicciones utilizando la aplicación. Estas imágenes se extrajeron del lote de pruebas utilizado durante la fase de entrenamiento de la CNN.

En conclusión, la prueba cerrada proporcionó retroalimentación con respecto al desarrollo de la interfaz, usabilidad y funcionalidad del prototipo. Esta retroalimentación fue esencial para realizar mejoras significativas en la aplicación. Los usuarios pudieron interactuar con la aplicación y proporcionar comentarios directos sobre su experiencia. Esto permitió identificar y corregir problemas, mejorar la interfaz de usuario y optimizar la funcionalidad de la aplicación.

5.5.4. Lanzamiento Global del Prototipo en Google Play

Tras la conclusión de la fase de prueba cerrada, Google Play autorizó el despliegue global de la aplicación. Esto significa que la aplicación ahora está disponible para ser descargada y utilizada por cualquier persona en todo el mundo que tenga acceso a un dispositivo móvil con sistema operativo *Android*.

Además, es importante mencionar que la aplicación ha sido optimizada para ser utilizada en una amplia gama de dispositivos *Android*. La aplicación cuenta con una interfaz de usuario intuitiva y fácil de usar, lo que facilita a los usuarios la realización de predicciones precisas sobre las tilapias utilizando las imágenes proporcionadas, esto gracias a los comentarios realizados en la prueba cerrada. Finalmente, se ha establecido un correo electrónico de soporte al cliente para atender cualquier consulta o problema que puedan tener los usuarios.

5.6. Pruebas de campo

En la fase final del desarrollo del prototipo de la aplicación web para la clasificación visual de sexo de tilapias por medio de las papilas genitales, se realizaron una serie de pruebas de campo en cultivos de tilapia del Nilo. El objetivo de estas pruebas era evaluar la funcionalidad de la aplicación en un entorno real y probar la exactitud del modelo CNN utilizado. Para ello, tres personas con conocimiento intermedio en la técnica de clasificación visual participaron y en paralelo se hicieron las pruebas correspondientes con la aplicación, con el objetivo de verificar la precisión y efectividad de la herramienta en la identificación correcta del sexo de las tilapias, los detalles pueden se muestran en el Apéndice D.

Las pruebas fueron realizadas por personal de la Universidad del Mar en sus instalaciones. Se utilizaron tres dispositivos diferentes: un POCO F3 con una cámara principal trasera de 48 megapíxeles, un Redmi 10C con una cámara trasera de 50 megapíxeles, y un Galaxy A34 con una cámara principal trasera de 48 megapíxeles, cuyas matrices de confusión se muestra en las figuras 5.14 y 5.15, respectivamente. Para estas pruebas se capturaron un total de 108 imágenes.

Las métricas de exactitud, sensibilidad, especificidad, precisión, F1-score y MCC se muestran en la Tabla 5.4. Se observa que el prototipo de la aplicación móvil AlEd

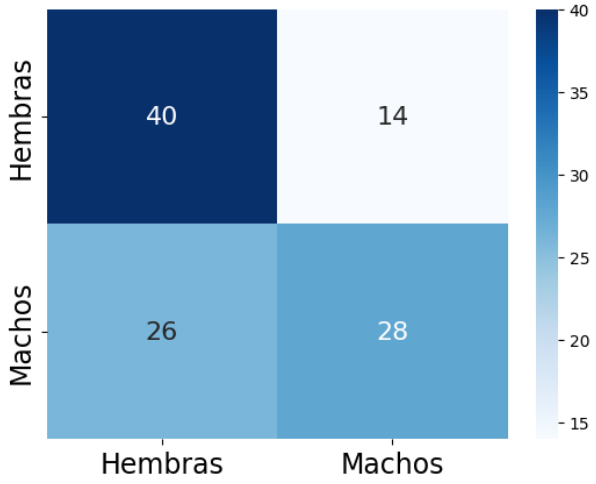


Figura 5.14: Matriz de confusión de los resultados obtenidos por medio de una persona con conocimiento intermedio en el área.

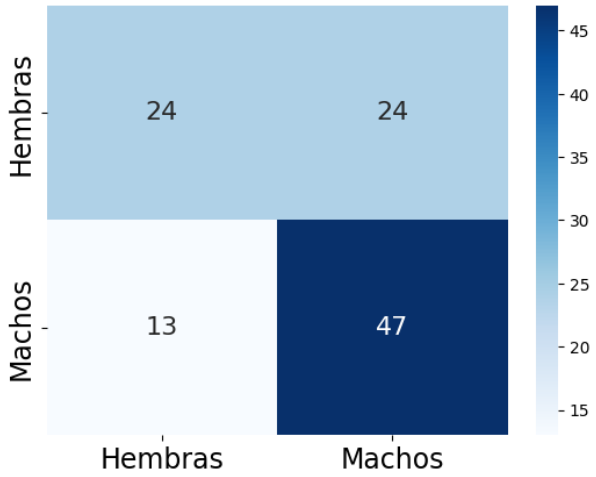


Figura 5.15: Matriz de confusión de los resultados obtenidos por medio del prototipo de aplicación móvil.

	Exactitud	**Sensitividad**	**Especificidad**	**Precisión**	**F1-Score**	**MCC**
Persona	0.6296	0.7407	0.5185	0.6060	0.6666	0.2659
AlEd	0.6574	0.5	0.7833	0.6486	0.5647	0.29666

Tabla 5.4: Comparación de métricas de clasificación entre una persona y el prototipo de aplicación móvil *AlEd*.

supera en promedio a las personas en cuanto a exactitud, especificidad, precisión, F1-score y MCC. Sin embargo, la sensibilidad es mayor en el caso de la persona. Estos resultados indican que el prototipo muestra un rendimiento superior a la clasificación visual realizada por un experto en el área, lo cual puede ofrecer una solución más completa y efectiva, en especial en caso cuando no hay un experto en el área.

Capítulo 6

Conclusión

En conclusión, este trabajo ha representado un esfuerzo con el objetivo de abordar los desafíos en la clasificación por sexo de tilapias del Nilo en la industria acuícola. A lo largo de este trabajo, se ha logrado avanzar en varios aspectos clave, ya que mediante una colaboración con investigadores de la Universidad del Mar, se logró construir un conjunto de datos compuesto por 3350 imágenes de papilas genitales de tilapias del Nilo. Este conjunto de datos puede ser de gran utilidad para otros investigadores en estudios relacionados.

En el proceso de revisión del estado del arte, se identificaron las arquitecturas *VGG-16* y *LeNet-5*, las cuales han sido ampliamente usadas en el ámbito del aprendizaje automático. En las pruebas realizadas en este estudio, la arquitectura *VGG-16* demostró un rendimiento de clasificación superior. Además, se llevó a cabo un ajuste de hiperparámetros para encontrar la configuración óptima del modelo que maximice su capacidad de generalización con el conjunto de datos obtenidos. El uso de validación cruzada permitió obtener un modelo de Red Neuronal Convolucional (CNN) más robusto e incrementar su rendimiento.

Una vez optimizado el modelo, se pudo exportar para su uso en el marco de tra-

bajo (*framework*) *React Native*, que está orientado a dispositivos móviles y sigue el paradigma de componentes. Este paradigma permite construir aplicaciones a partir de piezas de software reutilizables y autónomas, llamadas componentes. Esto permitió la creación de un prototipo de aplicación móvil. Utilizando *CSS* y *TypeScript*, y siguiendo el paradigma de componentes, se desarrolló una interfaz de usuario basada en las especificaciones del Capítulo 3. Esta interfaz permite interactuar con el hardware del dispositivo móvil y el modelo previamente entrenado, brindando al usuario final la capacidad de utilizar el prototipo para la clasificación de tilapias en campo.

A partir de la revisión del estado del arte, no se encontró ningún trabajo relacionado con la clasificación por sexo de tilapias del Nilo usando visión por computadora. Por lo tanto, este trabajo es de gran importancia para su aplicación en la industria de la acuicultura.

En el presente estudio se planteó la hipótesis de que la incorporación de CNN en un prototipo de aplicación móvil tendría el potencial de mejorar significativamente la precisión en la identificación sexual de tilapias, en comparación con los métodos tradicionales de clasificación manual realizados por expertos. Los resultados obtenidos respaldan la hipótesis planteada. El modelo basado en CNN logró una exactitud superior a la observada en estudios previos en el estado del arte, los cuales reportan rangos de precisón entre el 80 % y el 90 % [Penman and McAndrew, 2000, El-Sayed, 2006]. En contraste el modelo desarrollado en esta investigación alcanzó una exactitud del 98.33 % sobre el conjunto de prueba. Al comparar el rendimiento de este modelo con el de 10 expertos usando imágenes del conjunto de prueba enfocadas en las papilas genitales se observó que las personas expertas alcanzaron un promedio de exactitud del 76.90 %, otro conjunto de 27 personas con nivel de experiencia intermedia en el área obtuvo un 70.55 % de exactitud y el último grupo de 11 principiantes obtuvo un 62.04 % (ver Apéndice C). Finalmente, como se mostró en la Sección 5.6 los resultados en campo en condiciones reales, en donde intervinieron personas de nivel de experiencia intermedia tuvieron un 62.96 % de exactitud contra un 65.74 % usando la aplicación móvil (ver Apéndice D). Las pruebas de campo y el Análisis de Varianza indican que el prototipo presenta un nivel intermedio de identificación del sexo de la

tilapa, con un porcentaje mayor sobre el resultado con personas de nivel de experiencia intermedia (ver Apéndice E). Todas estas pruebas ponen en ventaja al modelo frente a los métodos tradicionales llevados a cabo por personas con diferente nivel de experiencia. Es importante aclarar que en las pruebas con condiciones reales pueden intervenir factores como la iluminación, el enfoque de la cámara del dispositivo y la experiencia de los usuarios al usar esta aplicación móvil, ya que esta es la primera vez que ellos la ocuparon en campo.

Es importante destacar que la aplicación no solo se destaca por su exactitud, sino también por su accesibilidad y facilidad de uso. Su interfaz intuitiva la hace adecuada para una amplia gama de usuarios, desde personal técnico hasta aquellos con menos experiencia en el campo de la acuicultura. Además, la aplicación no requiere conexión a internet, lo que facilita su uso en zonas remotas donde pueden encontrarse productores de tilapias con recursos limitados. El tiempo de respuesta rápido, con predicciones obtenidas en menos de dos segundos, agiliza los procesos de clasificación en campo, lo que mejora la eficiencia y la productividad en las operaciones acuícolas.

Además, es importante mencionar que se realizaron pruebas en campo a través de una prueba cerrada solicitada por Google Play. Estas pruebas permitieron validar la funcionalidad y obtener retroalimentación de los usuarios de la aplicación en condiciones reales de uso. Esta combinación de exactitud, facilidad de uso, funcionalidad sin conexión y validación en campo hace que la aplicación sea una herramienta valiosa y práctica para la industria, especialmente en áreas con acceso limitado a la conectividad en línea.

6.1. Trabajos a futuro

En cuanto a trabajos futuros, se contempla la posibilidad de que la aplicación evolucione para capturar video en tiempo real y realizar la clasificación de tilapias aplicando los modelos desarrollados en este estudio. Esta expansión implica el desa-

rrollo de hardware embebido especializado que pueda operar dentro de los entornos de cultivo de tilapias.

Es importante destacar que este desarrollo ha logrado acercar la herramienta de clasificación a personas que no tienen un entrenamiento previo para determinar el sexo de las tilapias. Esto puede apoyar el manejo del cultivo y obtener mejores rendimientos productivos. Además, este avance ha contribuido a hacer la actividad más sustentable ambientalmente al reducir el uso de químicos como el azul de metileno, comúnmente empleado en la clasificación tradicional de tilapias para resaltar las papilas genitales.

Aunado a esto, se espera que a través de un convenio establecido, el personal de la Universidad del Mar (UMAR) lleve a cabo más pruebas que permitan validar y profundizar en los resultados obtenidos en este estudio. Estas pruebas adicionales podrían proporcionar una base sólida para la publicación de los hallazgos, contribuyendo así al cuerpo de conocimiento existente en el campo de la acuicultura y la visión por computadora.

Bibliografía

[Albawi et al., 2017] Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

[Alcántar-Vázquez et al., 2015] Alcántar-Vázquez, J. P., Rueda-Curiel, P., Calzada-Ruíz, D., Antonio-Estrada, C., and Moreno-de la Torre, R. (2015). Feminization of the nile tilapia oreochromis niloticus by estradiol-17β effects on growth, gonadal development and body composition. *Hidrobiológica*, 25(2):275–283.

[Almero et al., 2019] Almero, V. J., Concepcion, R., Rosales, M., Vicerra, R. R., Bandala, A., and Dadios, E. (2019). An aquaculture-based binary classifier for fish detection using multilayer artificial neural network. In *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–5. IEEE.

[Apple, 2024] Apple (2024). Swift - apple developer. https://developer.apple.com/swift/. Última vez visitado Marzo-2024.

[Beveridge and McAndrew, 2012] Beveridge, M. C. and McAndrew, B. (2012). *Tilapias: biology and exploitation*, volume 25. Springer Science & Business Media.

[Bhujel, 2014] Bhujel, R. C. (2014). *A manual for tilapia business management*. CABI.

[Camacho Escobar, 2023] Camacho Escobar, M. A., V. A. H. (2023). *Zootecnia de especies alternativas en Oaxaca, México.* Universidad del Mar.

[Chollet, 2021] Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

[Dekanovský, 2020] Dekanovský, V. (2020). Machine-leaning-in-examples/sklearn/cross-validation/cross validation.ipynb at master · vaasha/machine-leaning-in-examples. https://github.com/vaasha/Machine-leaning-in-examples/blob/master/sklearn/cross-validation/Cross%20Validation.ipynb. Última vez visitado Marzo-2024.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Developers, 2024] Developers, A. (2024). El formato android app bundle — android developers. https://developer.android.com/guide/app-bundle/app-bundle-format?hl=es-419. Última vez visitado Marzo-2024.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[El-Sayed, 2006] El-Sayed, A.-F. M. (2006). *Tilapia culture*. CABI publishing.

[Expo, 2023] Expo (2023). Imagemanipulator - expo documentation. https://docs.expo.dev/versions/latest/sdk/imagemanipulator/. Última vez visitado Octubre-2023.

[Expo, 2024a] Expo (2024a). Create your first build - expo documentation. https://docs.expo.dev/build/setup/. Última vez visitado Marzo-2024.

[Expo, 2024b] Expo (2024b). Eas build - expo documentation. https://docs.expo.dev/build/introduction/. Última vez visitado Marzo-2024.

[Expo, 2024c] Expo (2024c). Expo. https://expo.dev/. Última vez visitado Marzo-2024.

[FAO, 2023] FAO (2023). Fisheries and aquaculture. https://www.fao.org/fishery/en/aquaculture. Última vez visitado Febrero-2023.

[Fernandes et al., 2020] Fernandes, A. F., Turra, E. M., de Alvarenga, É. R., Passafaro, T. L., Lopes, F. B., Alves, G. F., Singh, V., and Rosa, G. J. (2020). Deep learning image segmentation for extraction of fish body measurements and prediction of body weight and carcass traits in nile tilapia. *Computers and electronics in agriculture*, 170:105274.

[Fitch, 1944] Fitch, F. B. (1944). Warren s. mcculloch and walter pitts. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133. *The Journal of Symbolic Logic*, 9(2):49–50.

[Gong and Kan, 2021] Gong, Z. and Kan, L. (2021). Segmentation and classification of renal tumors based on convolutional neural network. *Journal of Radiation Research and Applied Sciences*, 14(1):412–422.

[Gonzalez, 2009] Gonzalez, R. C. (2009). *Digital image processing*. Pearson education india.

[Hahnloser et al., 2000] Hahnloser, R. H., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., and Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature*, 405(6789):947–951.

[Hasan et al., 2022] Hasan, N., Ibrahim, S., and Aqilah Azlan, A. (2022). Fish diseases detection using convolutional neural network (cnn). *International Journal of Nonlinear Analysis and Applications*, 13(1):1977–1984.

[Hawkins, 2004] Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

[Haykin, 2009] Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.

[Hernandez and Hernandez, 2019] Hernandez, R. M. and Hernandez, A. A. (2019). Classification of nile tilapia using convolutional neural network. In *2019 IEEE 9th*

International Conference on System Engineering and Technology (ICSET), pages 126–131. IEEE.

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574.

[Kelly, 2001] Kelly, R. (2001). Donnie darko. Film.

[Keras, 2023] Keras (2023). Keras: Deep learning for humans. https://keras.io. Última vez visitado Septiembre-2023.

[Keras, 2024] Keras (2024). Keras applications. https://keras.io/api/applications/. Última vez visitado Marzo-2024.

[Khan et al., 2018] Khan, S., Rahmani, H., Shah, S. A. A., and Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis lectures on computer vision*, 8(1):1–207.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kinser, 2018] Kinser, J. M. (2018). *Image Operators: Image Processing in Python*. CRC Press.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Li et al., 2021] Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*.

[Luque et al., 2019] Luque, A., Carrasco, A., Martín, A., and de Las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231.

[Martínez-Cordero et al., 2021] Martínez-Cordero, F. J., Delgadillo, T., Sanchez-Zazueta, E., and Cai, J. (2021). *Tilapia Aquaculture in Mexico-Assessment with a focus on social and economic performance*. Food & Agriculture Org.

[Matt Moore, 2024] Matt Moore, M. H. (2024). Tensorflow-kotlin/tensorflow-kotlin: Tensorflow api for kotlin. https://github.com/TensorFlow-Kotlin/tensorflow-kotlin. Última vez visitado Marzo-2024.

[Meyer et al., 2015] Meyer, D. et al. (2015). Reproducción y cría de alevines de tilapiamanual práctico.

[Muller and Guido, 2017] Muller, A. C. and Guido, S. (2017). *Introduction to machine learning with Python*. O'Reilly.

[Native, 2023a] Native, R. (2023a). React native · learn once, write anywhere. https://reactnative.dev/. Última vez visitado Septiembre-2023.

[Native, 2023b] Native, R. (2023b). Style react native. https://reactnative.dev/docs/style. Última vez visitado Octubre-2023.

[Navotas et al., 2018] Navotas, I. C., Santos, C. N. V., Balderrama, E. J. M., Candido, F. E. B., Villacanas, A. J. E., and Velasco, J. S. (2018). Fish identification and freshness classification through image processing using artificial neural network. *ARPN Journal of engineering and Applied Sciences*, 13(18):4912–4922.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Penman and McAndrew, 2000] Penman, D. and McAndrew, B. (2000). Genetics for the management and improvement of cultured tilapias. In *Tilapias: Biology and exploitation*, pages 227–266. Springer.

[Perschbacher and Stickney, 2017] Perschbacher, P. W. and Stickney, R. R. (2017). *Tilapia in intensive co-culture*. John Wiley & Sons.

[Play, 2024] Play, G. (2024). Requisitos de pruebas de apps para las cuentas de desarrollador personales nuevas - ayuda de play console. https://support.google.com/googleplay/android-developer/answer/14151465#production. Última vez visitado Abril-2024.

[Powell, 2023] Powell, V. (2023). Image kernels explained visually. https://setosa.io/ev/image-kernels/. Última vez visitado Octubre-2023.

[Prince, 2023] Prince, S. J. (2023). *Understanding Deep Learning*. MIT PRESS.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[Russell and Norvig, 2010] Russell, S. J. and Norvig, P. (2010). *Artificial intelligence a modern approach*. London.

[Shirley et al., 2009] Shirley, P., Ashikhmin, M., and Marschner, S. (2009). *Fundamentals of computer graphics*. AK Peters/CRC Press.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[StatSoft, Inc., 2011] StatSoft, Inc. (2011). Statistica (version 10). http://www.statsoft.com. Software estadístico.

[Sun et al., 2020] Sun, M., Yang, X., and Xie, Y. (2020). Deep learning in aquaculture: A review. *J. Comput*, 31(1):294–319.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[TensorFlow, 2023a] TensorFlow (2023a). Introducción a tensorflow. https://www.tensorflow.org/learn?hl=es-419. Última vez visitado Septiembre-2023.

[TensorFlow, 2023b] TensorFlow (2023b). Tensorflow.js — aprendizaje automático para desarrolladores. https://www.tensorflow.org/js?hl=es-419. Última vez visitado Septiembre-2023.

[TensorFlow, 2023c] TensorFlow (2023c). Tensorflow.js react native api. https://js.tensorflow.org/api_react_native/0.8.0/#bundleResourceIO. Última vez visitado Septiembre-2023.

[TensorFlow, 2024a] TensorFlow (2024a). Ejemplos de tensorflow lite — apps de aprendizaje automático para dispositivos móviles. https://www.tensorflow.org/lite/examples?hl=es-419. Última vez visitado Febrero-2024.

[TensorFlow, 2024b] TensorFlow (2024b). Save, serialize, and export models — tensorflow core. https://www.tensorflow.org/guide/keras/serialization_and_saving. Última vez visitado Febrero-2024.

[TensorFlow, 2024c] TensorFlow (2024c). tf.keras.preprocessing.image.image — tensorflow v2.15.0.post1. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator. Última vez visitado Enero-2024.

[Trejo-Quezada et al., 2021] Trejo-Quezada, A., Calzada-Ruiz, D., Soriano-Luis, F., Valenzuela-Jimenez, N., Ramirez-Ochoa, M., Torre, R. M.-d. l., and Alcántar-Vázquez, J. P. (2021). Evaluation of the masculinization period in the nile tilapia spring strain employing 17-methyltestosterone. *Ecosistemas y recursos agropecuarios*, 8(1).

[Vásquez-Quispesivana et al., 2022] Vásquez-Quispesivana, W., Inga, M., and Betalleluz-Pallardel, I. (2022). Artificial intelligence in aquaculture: basis, applications, and future perspectives.

[Zamir et al., 2018] Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722.

[Zar, 2010] Zar, J. H. (2010). *Biostatistical Analysis*. Pearson Prentice-Hall.

[Zhang et al., 2020] Zhang, L., Li, W., Liu, C., Zhou, X., and Duan, Q. (2020). Automatic fish counting method using image density grading and local regression. *Computers and Electronics in Agriculture*, 179:105844.

[Zion et al., 2008] Zion, B., Alchanatis, V., Ostrovsky, V., Barki, A., and Karplus, I. (2008). Classification of guppies'(poecilia reticulata) gender by computer vision. *Aquacultural Engineering*, 38(2):97–104.

Apéndice A

AlEd - Política de Privacidad

La política de privacidad permite informar al usuario de una aplicación móvil sobre cómo se recopilan, utilizan y protegen tus datos personales. Esta política es fundamental para garantizar su privacidad y seguridad en línea. Brinda transparencia sobre qué información se recopila, cómo se utiliza y si se comparte con terceros. Enseguida se listan los puntos que abarca la Política de Privacidad del prototipo publicada en la tienda de *Google Play*:

A.1. Política de privacidad

El presente Política de Privacidad establece los términos en que AlEd usa y protege la información que es proporcionada por sus usuarios al momento de utilizar la aplicación móvil AlEd. Esta aplicación está comprometida con la seguridad de los datos de sus usuarios. Cuando solicitamos acceso a la cámara, no recopilamos imágenes ni datos personales de ningún tipo. Esta Política de Privacidad puede cambiar con el tiempo o ser actualizada, por lo que se recomienda revisar continuamente esta página para asegurarse de que está de acuerdo con dichos cambios.

A.2. Información que es recogida

AlEd no recoge ni solicita ninguna información personal, incluidos nombres, direcciones de correo electrónico o información demográfica. Únicamente se solicita acceso a la cámara para el propósito exclusivo de permitir al usuario tomar fotografías de tilapias del Nilo para su clasificación sexual.

A.3. Uso de la información recogida

La aplicación AlEd no recoge ni almacena información de ningún tipo. Por lo tanto, no se emplea ninguna información para mantener registros de usuarios o mejorar productos y servicios.

A.4. Cookies

La aplicación AlEd no utiliza cookies ni herramientas de seguimiento similares.

A.5. Enlaces a Terceros

La aplicación AlEd no contiene enlaces a terceros.

A.6. Control de su información personal

Dado que AlEd no recoge información personales, no hay necesidad de restringir la recopilación o el uso de información personal. La aplicación no envía correos electrónicos, boletines o publicidad.

Apéndice B

Resto de resultados del ajuste de hiperparámetros

Tasa de aprendizaje	Eliminación aleatoria (Dropout)	No. de épocas	Unidades	Precisión	Pérdida
0.0005	0.3	80	512	0.9542	0.1284
0.0005	0.3	80	1024	0.9583	0.0878
0.0005	0.3	100	512	0.9458	0.1412
0.0005	0.3	100	1024	0.9667	0.1278
0.0005	0.3	120	512	0.9458	0.1498
0.0005	0.3	120	1024	0.9458	0.1423
0.0005	0.4	80	512	0.9625	0.1634
0.0005	0.4	80	1024	0.9500	0.1395
0.0005	0.4	100	512	0.9375	0.1532
0.0005	0.4	100	1024	0.9542	0.1135
0.0005	0.4	120	512	0.9042	0.2901
0.0005	0.4	120	1024	0.9542	0.1486

Continúa en la siguiente página

Tabla B.1 – continuación de la página anterior

Tasa de aprendizaje	Eliminación aleatoria (Dropout)	No. de épocas	Unidades	Precisión	Pérdida
0.0005	0.5	80	512	0.9500	0.2043
0.0005	0.5	80	1024	0.9333	0.1416
0.0005	0.5	100	512	0.9625	0.1184
0.0005	0.5	100	1024	0.9417	0.1663
0.0005	0.5	120	512	0.9500	0.1245
0.0005	0.5	120	1024	0.9167	0.1969
0.0025	0.3	80	512	0.5000	0.6932
0.0025	0.3	80	1024	0.5000	0.6932
0.0025	0.3	100	512	0.5000	0.6932
0.0025	0.3	100	1024	0.5000	0.6932
0.0025	0.3	120	512	0.5000	0.6932
0.0025	0.3	120	1024	0.5000	0.6931
0.0025	0.4	80	512	0.5000	0.6931
0.0025	0.4	80	1024	0.5000	0.6931
0.0025	0.4	100	512	0.5000	0.6932
0.0025	0.4	100	1024	0.5000	0.6931
0.0025	0.4	120	512	0.5000	0.6931
0.0025	0.4	120	1024	0.5000	0.6931
0.0025	0.5	80	512	0.5000	0.6932
0.0025	0.5	80	1024	0.5000	0.6931
0.0025	0.5	100	512	0.5000	0.6932
0.0025	0.5	100	1024	0.5000	0.6931
0.0025	0.5	120	512	0.9542	0.1424
0.0025	0.5	120	1024	0.8917	0.2361

Tabla B.1: Resultados de las 32 configuraciones obtenidas en la búsqueda y ajuste de hiperparámetros.

Apéndice C

Pruebas del conjunto de datos con personas de diferente nivel de experiencia

Con el objetivo de establecer un marco de comparación entre los humanos de diferentes niveles de experiencia y la aplicación, se generó un conjunto de datos compuesto por imágenes de las papilas genitales de estos peces, estas imágenes fueron tomadas del conjunto de prueba. Para evaluar la eficacia y la precisión del conjunto de datos, se diseñó un cuestionario en una página web, en el cual se presentaron las imágenes a 48 usuarios con diferentes niveles de experticia en la identificación de tilapias.

Estos usuarios fueron invitados a clasificar los especímenes basándose en las imágenes de las papilas genitales. Los resultados de esta prueba fueron cuidadosamente registrados y analizados, presentándose en tablas C.1 C.2 y C.3 que reflejan el desempeño de los usuarios según su nivel de experiencia. Estas tablas proporcionan una visión detallada de cómo la experiencia influye en la precisión de la clasificación y permiten identificar patrones y áreas de mejora en el proceso de identificación

Expertos			
Num.	Exactitud	Num.	Exactitud
1	73.0 %	6	68.0 %
2	80.5 %	7	78.0 %
3	78.0 %	8	75.5 %
4	65.5 %	9	82.5 %
5	83.0 %	10	85.0 %
		Promedio	76.90 %

Tabla C.1: Resultados de la clasificación de imágenes realizada por 10 humanos expertos.

manual.

Principiantes			
Num.	Exactitud	Num.	Exactitud
1	75.0 %	7	65.0 %
2	60.0 %	8	50.0 %
3	67.5 %	9	57.5 %
4	37.5 %	10	85.0 %
5	65.0 %	11	42.5 %
6	77.5 %		
		Promedio	63.75 %

Tabla C.2: Resultados de la clasificación de imágenes realizada por humanos con nivel de experticia básica (principiantes).

Este enfoque no solo ayuda a validar el conjunto de datos generado, sino que también proporciona una base comparativa para futuras mejoras y desarrollo de herramientas automatizadas de clasificación. Al entender mejor las capacidades y limitaciones de los expertos humanos, se puede optimizar el entrenamiento de modelos de aprendizaje profundo, logrando una mayor precisión y fiabilidad en la clasificación de tilapias.

La Figura C.1 resume los resultados obtenidos por el cuestionario con humanos de diferente nivel de experticia.

Intermedios					
Num.	Exactitud	Num.	Exactitud	Num.	Exactitud
1	75.0 %	10	82.5 %	19	75.0 %
2	75.0 %	11	47.5 %	20	75.0 %
3	52.5 %	12	67.5 %	21	52.5 %
4	47.5 %	13	62.5 %	22	47.5 %
5	77.5 %	14	82.5 %	23	77.5 %
6	62.5 %	15	57.5 %	24	62.5 %
7	77.5 %	16	75.0 %	25	77.5 %
8	87.5 %	17	67.5 %	26	87.5 %
9	75.0 %	18	75.0 %	27	75.0 %
				Promedio	70.56 %

Tabla C.3: Resultados de la clasificación de imágenes realizada por humanos con nivel de experticia intermedia.

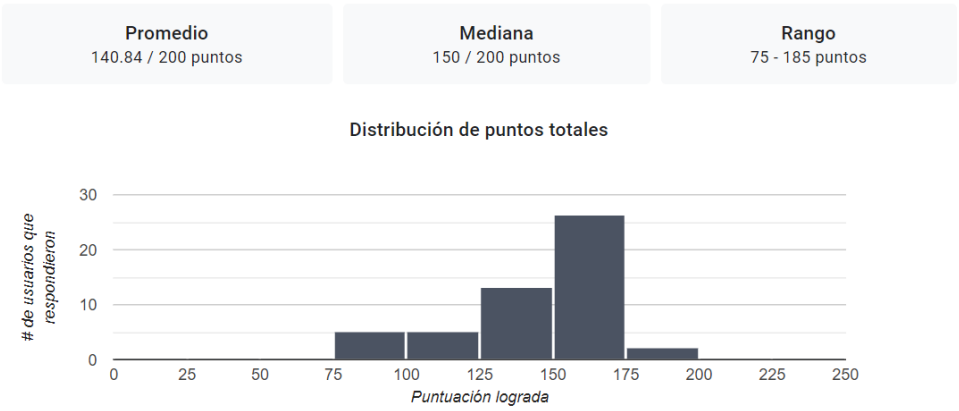


Figura C.1: Estadísticas de las pruebas hechas con humanos de diferente nivel de experticia.

Apéndice D

Resultados de pruebas en campo con personal de experiencia intermedia

El prototipo de aplicación móvil para la clasificación de tilapias por sexo está diseñado para capturar fotografías de tilapias y clasificarlas por sexo de manera automatizada usando un modelo de ML basado en CNN, con el objetivo de facilitar el trabajo de identificación en campo. Para evaluar su eficacia y precisión, se realizaron pruebas exhaustivas utilizando 36 especímenes de tilapias.

En la primera fase de las pruebas, participaron 3 usuarios con un nivel de experiencia intermedia que realizaron la clasificación visual de las tilapias. La primera columna de la Tabla D.1 muestra el sexo identificado a partir del sacrificio de los especímenes por punción craneal para confirmar su género de manera precisa por verificación de las gónadas, proporcionando una medida objetiva de la exactitud. Para verificar el sexo de los peces prueba, estos se sacrificaron por punción craneal, se realizó un corte longitudinal en la parte ventral y se extrajeron las gónadas, por lo que se identificaron 25 hembras y 11 machos. En paralelo, la aplicación móvil

fue utilizada para tomar fotografías enfocadas en sus papilas genitales y realizar la clasificación; las últimas 6 columnas muestran el resultado de esta clasificación y el porcentaje de precisión también mostrado por la aplicación.

Estas pruebas comparativas permiten analizar la precisión del sistema automatizado frente a la clasificación visual realizada por humanos con experiencia intermedia. Estos resultados muestran en promedio que la aplicación móvil tiene una ligera ventaja sobre estos usuarios en condiciones reales. Los resultados obtenidos no sólo proporcionan información valiosa sobre la efectividad de la aplicación móvil, sino que también destacan las posibles mejoras y ajustes necesarios para optimizar la herramienta.

Este enfoque integral busca ofrecer una solución práctica y confiable para la clasificación de tilapias en el ámbito acuícola, mejorando la eficiencia y reduciendo los errores asociados a la identificación manual.

Valor real	**Pers. 1**	**Pers. 2**	**Pers. 3**	**Teléf. 1**	**Teléf. (%) 1**	**Teléf. 2**	**Teléf. (%) 2**	**Teléf. 3**	**% Teléf. (%) 3**
H	H	H	H	M	79.49	H	85.20	H	98.98
H	H	H	H	H	49.99	H	61.41	H	73.25
M	M	M	M	M	100.00	M	89.66	H	96.98
H	M	H	H	M	73.25	M	53.30	H	99.91
H	M	M	H	M	99.31	H	52.53	H	93.59
H	M	M	M	M	57.32	H	98.75	H	74.25
H	H	H	M	M	53.99	H	96.37	M	58.60
H	M	M	M	M	92.22	H	98.35	H	76.60
M	M	H	M	M	100.00	M	93.78	H	70.94
H	H	M	M	H	59.23	M	70.13	H	98.02
H	M	M	M	M	85.07	H	98.83	H	82.58
H	H	H	H	M	95.78	H	82.56	M	50.88
H	M	M	M	M	95.91	H	88.45	H	51.93
H	H	M	H	M	70.96	H	99.89	H	83.35
M	M	M	M	M	90.08	M	91.12	H	99.50
M	M	H	H	M	99.47	M	55.75	H	95.27
H	H	H	H	H	100.00	H	100.00	H	100.00
H	H	M	H	M	69.66	H	87.16	H	96.88
M	M	M	M	H	99.76	H	99.85	H	96.76
M	M	H	M	M	100.00	M	99.96	M	63.49
M	M	M	M	M	89.78	M	99.19	M	50.90
H	H	M	M	H	70.14	H	65.22	H	72.24
H	H	M	H	H	97.00	H	95.71	H	99.56
H	H	M	M	H	82.86	H	99.80	H	67.74
H	H	H	H	H	81.11	M	59.90	H	97.92
H	H	H	H	M	96.26	H	59.76	M	67.95
H	M	M	M	H	97.70	H	99.53	H	97.77
H	H	H	H	M	91.12	M	61.50	M	63.87
M	M	M	M	M	100.00	M	88.95	M	98.32
H	H	M	H	M	67.03	H	99.14	H	99.44
H	H	H	H	M	97.27	M	69.61	M	88.07
M	M	M	M	M	76.78	M	61.93	M	66.49
H	M	M	M	H	91.64	H	99.57	H	99.09
H	H	H	H	M	89.00	M	70.59	M	64.71
M	H	M	M	M	88.61	M	99.86	M	65.87
M	H	H	H	M	74.68	H	70.97	H	63.98
Exactitud (%)	72.22	50.00	66.67	52.78		77.78		66.67	

Tabla D.1: Resultados de sexado manual y usando la aplicación de 36 ejemplares de tilapias.

Apéndice E

Evaluación del prototipo AlEd en campo

E.1. Diseño experimental

Para evaluar el prototipo de aplicación móvil AlEd para el sexado de tilapia en campo, se propuso un diseño experimental de una vía con dos tratamientos: Persona y Teléfono, cada uno con tres réplicas.

E.2. Materiales y métodos

Para la evaluación se seleccionaron al azar 36 ejemplares de tilapia, con un peso promedio 268.33±95.73 gr., de las tinas de cultivo de los Laboratorios de Acuicultura de la Universidad del Mar, campus Puerto Ángel.

Las personas que intervinieron en la prueba tenían un nivel de experiencia inter-

medio en el sexado de tilapia, y siguieron el procedimiento tradicional de observación de la papila genital sin tinte para determinar el sexo del pez.

Los teléfonos que se usaron fueron: marca Xiaomi modelo Redmi 10C, marca Samsung modelo Galaxy A34, y marca Xiaomi modelo Poco F3. El procedimiento se basó en enfocar con la cámara del teléfono la papila genital y realizar un aumento del área genital de ser necesario.

Para verificar el sexo de los peces prueba, estos se sacrificaron por punción craneal, por lo que se identificaron 25 hembras y 11 machos.

E.3. Recolección de datos

Se recabaron los datos obtenidos por las personas y los teléfonos y se compararon con el valor real del sexo de la tilapia para determinar aciertos y errores. Los aciertos tomaron el valor de 1 y se sumaron para obtener el porcentaje de aciertos.

E.4. Análisis estadístico

Se comprobaron los supuestos de normalidad y homocedasticidad de los datos con las pruebas de Kolmogorov-Smirnov y Levene. Para las pruebas de normalidad se utilizaron los residuales. Para determinar diferencias en los datos se utilizó un análisis de varianza (ANDEVA) de una vía. Todos los análisis se realizaron con una significancia nominal de 5 % [Zar, 2010], usando el programa Statistica 10 [StatSoft, Inc., 2011].

E.5. Resultados

El prototipo de aplicación móvil AlEd tuvo un porcentaje promedio de aciertos de 65.73 ± 12.52, y las personas de 62.96 ± 11.56. La prueba de ANDEVA indicó que no hubo diferencias significativas entre tratamientos (p= 0.79) (Tabla E.1).

	Univariate Results for Each DV (sexado) Sigma-restricted parameterization Effective hypothesis decomposition				
Effect	Degr. of Freedom	Porcentaje SS	Porcentaje MS	Porcentaje F	Porcentaje p
Intercept	1	24844.25	24844.25	171.0659	0.000197
Tratamiento	1	11.56	11.56	0.0796	0.791798
Error	4	580.93	145.23		
Total	5	592.49			

Tabla E.1: Resultados del Análisis de Varianza aplicado a los porcentajes de sexado de tilapia en la prueba de campo.

E.6. Conclusiones

El prototipo de aplicación móvil AlEd presenta un nivel intermedio de identificación del sexo de la tilapia en campo. A pesar de que no hubo diferencias estadísticas respecto al resultado con personas, el porcentaje promedio mayor observado en el prototipo móvil AlEd es un indicativo de que la aplicación se puede mejorar.