UNIVERSIDAD NACIONAL DE MISIONES
FACULTAD DE INGENIERÍA UNaM

Universidad Nacional de Misiones. Facultad de Ingeniería.
Carrera de Ingeniería en Computación

Estudiantes
De Sosa, Héctor
Zakowicz, Sandro Daniel

Machine Learning en Aplicación Móvil para la Clasificación del Estado de la Abeja Reina mediante Sonido

Informe de Proyecto Final Integrador presentado para obtener el título de "Ingeniero en Computación"

"Este documento es resultado del financiamiento otorgado por el Estado Nacional, por lo tanto, queda sujeto al complimiento de la Ley N° 26.899".

Director
Ing. Axel A. Skrauba

Oberá, Misiones, diciembre 2024

Universidad Nacional de Misiones

Facultad de Ingeniería

Informe de Proyecto Final Integrador

Machine Learning en Aplicación Móvil para la Clasificación del Estado de la Abeja Reina mediante Sonido

Para la obtención del título de

Ingeniero en Computación

Por

De Sosa, Héctor

Zakowicz, Sandro Daniel

Tutor:

Ing. Skrauba, Axel A.

Como miembros del Tribunal Examinador del Proyecto Final Integrador certificamos que hemos leído el documento del informe final preparado por el Sr. Héctor DE SOSA y el Sr. Sandro Daniel ZAKOWICZ, titulado "Machine Learning en Aplicación Móvil para la Clasificación del Estado de la Abeja Reina mediante Sonido" y recomendamos su aprobación.

Dr. Ing. Javier E. Kolodziej

Ing. Andrea G. Santander

Dra. Lic. Nancy B. Ganz

Oberá, [17] de [Diciembre] de [2024].

Índice

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

Agradecimientos

En primer lugar, agradecemos a Dios por habernos dado la fuerza y la sabiduría necesarias para llevar a cabo este proyecto, permitiéndonos cumplir con los objetivos propuestos y alcanzar esta etapa final en nuestra formación como ingenieros.

Expresamos nuestra gratitud a la Facultad de Ingeniería de Oberá por brindarnos la oportunidad de desarrollarnos académicamente y ser los primeros ingenieros en computación de la Universidad Nacional de Misiones, además, por brindarnos los laboratorios como espacios de trabajos para desarrollar este proyecto.

Un especial agradecimiento al Ing. Axel A. Skrauba, quien, como nuestro tutor, nos brindó una guía desde las primeras ideas hasta la concreción de

este trabajo. Su dedicación, paciencia y experiencia fueron pilares fundamentales en cada etapa del proyecto.

A nuestras familias, agradecemos inmensamente por el apoyo incondicional, tanto en lo emocional como en lo económico, y por estar presentes en cada desafío que enfrentamos.

También queremos reconocer a nuestros compañeros de cursada, quienes compartieron ideas y contribuyeron al desarrollo de nuestro proyecto.

Finalmente, agradecemos a la comunidad apícola y a todos aquellos investigadores que han dedicado su tiempo a estudiarla y compartir su conocimiento. Sin estos aportes, habría sido imposible culminar este proyecto, que busca contribuir con la apicultura misionera.

CAPÍTULO 2

Resumen

Este informe describe el desarrollo de "Queen Bee Monitor", una aplicación móvil para dispositivos Android que emplea *machine learning* para determinar el estado de la abeja reina en colmenas de Apis Mellífera mediante grabaciones de audio. Su propósito es agilizar y optimizar el proceso de comprobación de la abeja reina presente, así como mejorar la eficiencia en procedimientos para el cambio de reina. La clasificación de estados incluye: reina presente, reina ausente, reina rechazada y reina aceptada. La app desarrollada implementa dos modelos de *machine learning*. El primer modelo verifica si el audio proporcionado por el usuario corresponde a sonidos de abejas, y solo en caso afirmativo, el segundo modelo infiere el estado de la reina. La mayor parte de los datos de audio utilizados para el entrenamiento de los modelos son de

acceso público, disponibles en línea, con excepción de aquellos grabados en la zona. Se emplearon técnicas de remoción de *outliers*, corrección de etiquetas y balanceo de clases para mejorar la calidad de los datos. Las pruebas de campo confirmaron la utilidad de la aplicación para un diagnóstico rápido del estado de la abeja reina, reduciendo significativamente el tiempo requerido para determinar su presencia o ausencia. Esta herramienta facilita la gestión eficiente de las colmenas, especialmente durante el cambio de reina.

Abstract

This report describes the development of "Queen Bee Monitor", a mobile application for Android devices that uses machine learning to determine the status of the queen bee in Apis Mellifera hives through audio recordings. Its purpose is to streamline and optimize the process of verifying the presence of the queen bee, as well as to enhance efficiency in queen replacement procedures. The status classification includes: queen present, queen absent, queen rejected, and queen accepted. The developed app implements two machine learning models. The first model verifies whether the audio provided by the user corresponds to bee sounds, and only if confirmed, the second model infers the queen's status. Most of the audio data used for training the models is publicly accessible and available online, with the exception of recordings made in the local area. Techniques such as outlier removal, label correction, and class balancing were applied to improve data quality. Field tests confirmed the application's usefulness for rapid diagnosis of the queen bee's status, significantly reducing the time required to determine her presence or absence. This tool facilitates efficient hive management, especially during queen replacement.

Parte I

Contextualización

CAPÍTULO 3

Introducción

"Si las abejas desaparecieran del planeta, a los humanos sólo nos quedarían 4 años de vida."

Atribuido a Albert Einstein

Este trabajo aborda el proceso de desarrollo de "Queen Bee Monitor", una aplicación móvil para dispositivos Android enfocada en el sector apícola: está destinada a la clasificación del estado de la abeja reina mediante el sonido de la colmena.

La apicultura es una actividad que consiste en cuidar y manejar a las colmenas de abejas para obtener los diversos productos que estas ofrecen, así

como también los beneficios en la polinización de árboles y plantas. Se sabe que el ser humano aprovecha los productos derivados de las colmenas de abejas desde la antigüedad [1]. En el continente americano, las abejas europeas fueron introducidas en el siglo XVII. Luego, en 1956 se introdujeron las abejas africanas en Brasil, y se extendieron rápidamente a otros países [2]. Con el paso de los años se produjo un cruzamiento entre las abejas europeas y las africanas.

Una colmena es una estructura natural o artificial en la cual vive la colonia de abejas. Es común que los términos colmena y colonia se usen indistintamente. Las colonias de abejas están compuestas por tres tipos de integrantes que se diferencian por el rol. Hay miles de obreras, cientos de zánganos y una reina; esta última es fundamental para la supervivencia y productividad de la colonia, ya que es la única abeja que puede poner huevos fertilizados, lo que garantiza la continuidad de la misma. La presencia o ausencia de la reina puede indicar el estado de salud de la colonia. Si la reina está ausente o no es aceptada, la productividad y estabilidad de la colonia se ven comprometidas.

Tradicionalmente, los apicultores inspeccionan de manera manual las colmenas para verificar la presencia de la reina y el estado general de la colonia. Sin embargo, este proceso es invasivo por lo cual perturba a las abejas, lo que podría llevar a un estrés innecesario en la colonia y afectar negativamente la producción de miel debido a la manipulación de la colmena por el ser humano [3].

El uso del sonido de las abejas como herramienta para el estudio de las colmenas no es una práctica nueva, pero ha ganado relevancia significativa en las últimas décadas debido a los avances en tecnología de grabación y procesamiento de señales. Los primeros estudios se centraron en la observación del comportamiento de las abejas buscando correlaciones entre ciertos sonidos y el estado de la colmena. Estos métodos permitieron a los investigadores identificar patrones de zumbido que indicaban la presencia de la reina, enjambres inminentes [1], o estados de estrés dentro de la colonia [4], [5], [6].

[1] Se refiere a una situación en la que una colonia de abejas está cerca de dividirse y formar un enjambre

Para evitar estrés extra en las colmenas durante revisiones prolongadas, se propone un sistema que utiliza el análisis del sonido de las abejas como un indicador del estado de la colmena, empleando grabaciones breves de 10 a 120 segundos. Los cambios en los patrones de sonido revelan la presencia o ausencia de la reina, así como otros estados importantes de la colmena.

Este trabajo utiliza grabaciones de audio desde el interior de colmenas de Apis Mellifera para desarrollar modelos de aprendizaje automático (*machine learning*, ML) y su posterior integración en una aplicación móvil, con el fin de clasificar el estado de la abeja reina.

La importancia de este trabajo radica en una problemática a nivel local y otra a nivel global. A nivel global, se tiene la reducción del número de colmenas de abejas debido a un conjunto de diferentes factores como la contaminación, las plagas y las enfermedades. En este contexto, realizar una comprobación y seguimiento del estado de la abeja reina es crucial para mantener colmenas fuertes y resilientes. Sin ella, la población de obreras no se renueva y la colmena enfrenta un inminente colapso. Por otra parte, a nivel local, la actividad apícola está presente en toda la provincia de Misiones. Existen alrededor de 760 productores inscritos en el Registro Nacional de Productores Apícolas (RENAPA) [7], y debido a las condiciones de trabajo en los apiarios, surge la necesidad de contar con una herramienta que se pueda ejecutar en tiempo real pero sin acceso a internet.

En la ejecución de este proyecto se aplicaron competencias desarrolladas en inteligencia computacional, procesamiento digital de señales, ingeniería del software, programación, entre otras.

CAPÍTULO 4

Antecedentes

En las primeras décadas del siglo XXI, la mayoría de las áreas han evolucionado a causa de lo que se denomina la cuarta revolución industrial o era de la revolución digital. La apicultura también evolucionó de forma paulatina, aprovechando mejor las posibilidades que abren las nuevas tecnologías de la información [6], [8].

Uno de los primeros desarrollos ha sido BeeHome, desarrollado por la firma israelí Beewise [9]. Son estructuras en forma de contenedor que mediante robótica son capaces de llevar a cabo las tareas del apicultor, o en algunos casos, las de las propias abejas. En su interior pueden albergar hasta 64 colmenas que están continuamente monitoreadas con básculas, cámaras, micrófonos y otros sensores, ahorrando al apicultor ese trabajo. Si se produce

un cambio de temperatura o de peso en una colmena, la inteligencia artificial incorporada puede cambiar las condiciones de espacio y ventilación. Esto puede ocurrir, por ejemplo, para prevenir la enjambrazón [1]. BeeHome también puede iniciar una cosecha automática si la cantidad de miel es la apropiada y se encuentra en condiciones óptimas (madura). Además, es capaz de detectar la presencia de enfermedades o plagas, especialmente de varroa[2]. Además, toda la información se almacena y el apicultor puede recibirla y controlarla desde una aplicación en su teléfono o desde su ordenador.

A medida que la tecnología avanzó, también lo hicieron las aplicaciones del análisis de sonido en la apicultura. Hoy en día, se emplean técnicas sofisticadas de procesamiento de señales y ML para interpretar los datos de audio con mayor precisión y de forma automatizada. [6]. Asimismo, se destaca un estudio en el que se evaluó el comportamiento de las abejas ante la retirada y posterior reintroducción de una abeja reina [11]. Según los resultados que obtuvieron, en el caso de ausencia de la reina, se observa una gran amplitud de intensidad en las frecuencias bajas, las abejas cambian su nivel de actividad y están nerviosas, mientras que 9-10 días después de la reintroducción de la reina en la colmena, las colonias vuelven a sus sonidos normales.

Otro sistema desarrollado es Bee Sound Detector (BeeSD) [12], el cual incorpora sensores para la monitorización de sonido en tiempo real, con la combinación de sensores de temperatura, humedad y sonido, el BeeSD puede detectar eventos de trastorno de colapso de colonias debido a hambrunas y eventos climáticos extremos, pérdida de reinas y enjambres. Además, BeeSD utiliza el registro en la nube y una aplicación de teléfono móvil adecuada para enviar notificaciones de mediciones extremas a los agricultores. Con base en los indicadores de desempeño alcanzados, los autores presentan el funcionamiento de su dispositivo y sistema BeeSD IoT.

Por otro lado, la empresa OsBeeHives [13] emplea Inteligencia Artificial (IA) para interpretar el sonido de la colonia e informar el estado de

[1] Fenómeno natural por medio del cual las colmenas se dividen en dos y parte de las abejas se marchan con la reina.

[2] La varroa es un ácaro parásito de las abejas melíferas [10].

salud: colmena sana o debilitada. Se enfoca en detectar aquellos sonidos que las abejas emiten ante un agente agresor, como la varroa. Cuando una colonia está infestada por los ácaros varroa, producen una vibración muy específica que se puede aislar para detectar la presencia del ácaro [8].

Otro trabajo que se tuvo en cuenta, es el desarrollo de un algoritmo para la clasificación basada en sonido de la actividad de las abejas melíferas que informa una comparación preliminar entre varias características extraídas utilizadas por separado como entrada para un clasificador de red neuronal convolucional [14]. En particular, se ha considerado la situación de las colonias huérfanas utilizando un conjunto de datos adquiridos en una situación real. Se han llevado a cabo experimentos con diferentes configuraciones para probar el rendimiento del sistema propuesto y los resultados han confirmado su potencialidad.

En el campo del procesamiento de audio, la extracción de características es un paso fundamental que condiciona el desempeño del modelo de clasificación. Diversos métodos y algoritmos están disponibles para extraer características relevantes de señales de audio [5]. Cada uno con sus propias ventajas y desventajas. Entre los métodos más comunes se encuentran los Coeficientes Cepstrales en las Frecuencias de Mel (Mel Frequency Cepstral Coefficients- MFCC) y la Transformada Rápida de Fourier (Fast Fourier Transform- FFT), y recientemente, técnicas especializadas en la captura de descriptores en general, que se sustentan en las representaciones vectoriales densas (*embeddings*) de modelos profundos (*deep learning*), entrenados para tareas generales. En el caso concreto de audio, un ejemplo de estos modelos es YAMNet [15].

Los MFCCs han sido ampliamente adoptados en aplicaciones de reconocimiento de voz y han demostrado ser efectivos en el análisis de audio en apicultura, gracias a su capacidad para capturar la envolvente espectral de las señales de zumbido, un aspecto fundamental para diferenciar estados en la colmena [6]. En contraste, la FFT ofrece una representación frecuencial directa de la señal, lo que puede ser útil para identificar patrones específicos en el zumbido de las abejas. No obstante, en este estudio se prefieren los MFCCs

debido a su eficacia en el modelado de señales complejas, como las generadas por las abejas, lo que justifica su selección como método principal para la extracción de características.

Existen muchas aplicaciones móviles diseñadas para la apicultura. Entre las que se basan en ML se encuentran BeeScanning [16], ZiBees [17] o Varroa Counter [18], que emplean la cámara del teléfono para la detección automática de varroa. También existen apps para encontrar la reina empleando la cámara. Una de ellas es Bee Queen Detector [19]. Otras aplicaciones se centran en interpretar el sonido de las colmenas para hacer un diagnóstico de su estado. Entre ellas se encuentra Bee Health Guru [20], una app que realiza un análisis de la colmena simplemente "escuchando" su sonido durante medio minuto, y determina si la colmena está sana o enferma.

A partir de los antecedentes presentados, se puede apreciar que tanto el ML, como los sistemas de sensores y las aplicaciones móviles, están facilitando tareas como el monitoreo remoto de colmenas, la prevención de enfermedades, la detección de eventos, brindando a los apicultores nuevas herramientas que contribuyen a la mejora de la productividad.

En este contexto, el presente proyecto busca hacer una contribución significativa a la apicultura para la provincia de Misiones, Argentina, desarrollando una aplicación móvil llamada Queen Bee Monitor, que implementa ML para la clasificación del estado de la abeja reina de la especie Apis Mellifera en 4 estados posibles: original, ausente, rechazada y aceptada.

CAPÍTULO 5

Planteamiento del Problema

En la apicultura existen diversas tareas de manejo del apiario [1], y una de ellas es la comprobación del estado de la abeja reina. Esta última es indispensable para la supervivencia de la colonia. El problema detectado consiste en determinar el estado de la abeja reina de manera rápida sin provocar un estrés en la colmena. Para abrir una colmena se debe ahumar [2], esto provoca que las abejas abandonen su rutina habitual, y comiencen a tomar miel, preparándose instintivamente para huir en caso de que se acerque un incendio. De este modo se reduce su capacidad para picar y su agresividad. Si es lo único que se hace, la interrupción dura unas horas hasta que todas las abejas descarguen la miel y

[1] Es el lugar donde se concentran todas las colmenas en las que habitan las abejas [21].
[2] Introducir humo en la colmena para que las abejas se vuelvan temporalmente más dóciles.

vuelvan a sus tareas normales [22]. Sin embargo, buscar a la abeja reina implica remover y revisar los panales uno por uno. En el peor de los casos, se remueven todos los panales. El tiempo requerido para este procedimiento puede variar entre unos 30 minutos a más de 1 hora, dependiendo de varios factores como el tamaño de la colmena, la cantidad de alzas y cuadros, el aspecto de la reina, la cantidad de zánganos, entre otros. Sumado al estrés propio de abrir la colmena, está el ataque de otras colmenas cercanas que buscarán robar miel. Por lo tanto, la velocidad en este procedimiento es crucial. En resumen, existen cuatro escenarios principales en los cuales es interesante conocer el estado de la reina en una colmena de abejas Apis Mellifera:

- Reemplazo inducido de reina: cuando la abeja reina es muy vieja (más de 4 años), el apicultor debe matarla para que en la colmena surja una nueva reina más fértil. Una abeja reina tarda 16 días en nacer. Pasado ese lapso es importante verificar la presencia de la nueva reina.

- Inserción de nueva reina: cuando se introduce una nueva abeja reina ya fecundada, debe ser aceptada por la colonia. El procedimiento teórico indica que antes de abrir la colmena para comprobar si la reina fue aceptada, se debe esperar 10 días. No se recomienda abrir el cajón antes del plazo mencionado dado que se corre el riesgo de que las abejas rechacen a la reina nueva por ese motivo. Por lo tanto, hasta entonces se desconoce si la reina nueva fue aceptada.

- Enjambrazón: cuando se produce una enjambrazón, la reina de mayor edad es la que abandona la colmena, quedando las abejas reina jóvenes o que aún no han nacido. Estas son las que compiten por el puesto de reina. Es posible que debido a la competencia todas las reinas nuevas mueran, o que la nueva reina muera durante su vuelo nupcial [3] antes de regresar a la colonia.

[3] El primer vuelo de la abeja reina en el cual es fecundada por los zánganos antes de regresar a la colmena.

- Trasiego [4]: cuando los apicultores obtienen una nueva colonia silvestre y la introducen en una colmena artificial, es crucial que la reina se encuentre allí. A veces no se ve a la reina y se procede a trasladar a la mayor cantidad de abejas al nuevo cajón, esperando que la reina también esté allí.

En base a lo descrito anteriormente, un apicultor puede optar por abrir la colmena y buscar a la reina cuadro por cuadro, pero ello implica que las abejas obreras pierden energía, alimento y tiempo. Sumado a esto el riesgo de que abejas de otras colmenas ataquen y roben la miel.

Por último, si bien el uso de teléfonos inteligentes está extendido en la provincia, la conectividad a internet en un los apiarios ubicados en zonas rurales suele ser muy precaria, y las redes WiFi no tienen el alcance necesario. Este es un factor importante a la hora de desarrollar una aplicación móvil para el sector apícola.

[4]Operación de paso de una colmena natural (silvestre) o rústica a un cajón (colmena estándar) [23].

CAPÍTULO 6

Justificación

En apicultura, es esencial que las abejas trabajen de manera continua y sin gastar energía innecesaria. Contar con una herramienta de diagnóstico no invasiva para evaluar el estado de la abeja reina en la colmena resulta de gran utilidad. Con este fin, se desarrolla la aplicación Queen Bee Monitor, que permitirá a los apicultores optimizar su tiempo mediante un diagnóstico rápido del estado de la reina a través del sonido, sin causar estrés a la colonia ni interrumpir el trabajo de las abejas. Al ser no invasiva y de rápida implementación, esta herramienta contribuye a preservar el rendimiento de la colmena. Además, el uso de un dispositivo móvil elimina la necesidad de adquirir hardware adicional, convirtiendo la solución en una alternativa accesible y atractiva para los productores apícolas de la región.

La provincia de Misiones de la República Argentina cuenta con una actividad apícola significativa, favorecida por su abundante y variada floración durante gran parte del año [24]. La apicultura está presente en los 17 departamentos del distrito provincial, con 13.760 aproximadamente colmenas registradas en más de 890 apiarios, manejados por 760 productores inscritos en el Registro Nacional de Productores Apícolas (RENAPA) [7]. En total, la provincia produce más de 200.000 kg de miel anuales. A nivel nacional, existen 15.306 apicultores registrados en RENAPA, y con sus 3.548.894 colmenas, Argentina produce cerca de 76.000 toneladas de miel anualmente [25].

Dado que las abejas juegan un rol crucial en la biodiversidad y que el sector apícola tiene gran relevancia económica tanto a nivel provincial como nacional, es fundamental implementar tecnologías que automaticen procesos de manejo en apicultura. Esto no solo ahorra tiempo y recursos, sino que facilita el cuidado de las colmenas. Además, la disponibilidad de conjuntos de datos (*datasets*) creados ante la reducción global de colmenas, como consecuencia de factores como la contaminación, ha impulsado el desarrollo de soluciones tecnológicas para monitorear la salud de las abejas.

Queen Bee Monitor pretende reducir el tiempo necesario para comprobar el estado de la abeja reina, para cualquiera de los escenarios descritos anteriormente: reemplazo inducido de reina, inserción manual, enjambrazón o trasiego. Esto implica, además, una reducción del estrés provocado a la colmena, y una reducción del tiempo para realizar el seguimiento de las colmenas de un apiario.

Este proyecto cuenta con los recursos tecnológicos y el conocimiento necesarios para llevarse a cabo, incluyendo dos conjuntos de datos de audio de abejas que permitirán entrenar el modelo de clasificación. Asimismo, se disponen de colmenas donde realizar pruebas de campo para la validación de la aplicación.

En el futuro, es posible incorporar funciones adicionales a la aplicación para optimizar aún más los procesos en los apiarios y asistir a los apicultores en otras áreas clave de la gestión apícola.

CAPÍTULO 7

Objetivos

7.1. Objetivo General

Desarrollar una aplicación móvil que utilice *machine learning* para analizar sonido de colmenas y así determinar el estado de la abeja reina.

7.2. Objetivos Específicos

- Analizar y procesar los datos de audio, para obtener características relevantes que permitan el entrenamiento del modelo de *machine learning*.

- Desarrollar un modelo clasificador a partir de las características extraídas del *dataset* de audio para la inferencia del estado de la reina.
- Diseñar y programar la aplicación móvil utilizando el lenguaje Python y el *framework* Kivy
- Integrar el modelo de *machine learning* con la aplicación y compilarla para su ejecución en dispositivos con sistema operativo Android.
- Validar el funcionamiento de la aplicación mediante pruebas *in situ*.
- Evaluar el desempeño de la aplicación desarrollada y proponer mejoras futuras.

CAPÍTULO 8

Marco Teórico

8.1. Machine Learning

8.1.1. Definición

El *machine learning* (ML) puede definirse como el uso de un conjunto de algoritmos para analizar datos, aprender de ellos, y entonces determinar o predecir un conjunto de eventos [26]. Se trata de una de las ramas de la Inteligencia Artificial (IA), que es un campo que se centra en el desarrollo de procesos y mecanismos para que las máquinas imiten funciones cognitivas que solemos asociar con el comportamiento humano [27].

8.2. Análisis de audio

El sonido es una onda de presión longitudinal formada por las compresiones y rarefacciones de las moléculas de aire, que se propaga en dirección paralela a la aplicación de energía [28]. Cabe aclarar que cuando una onda sonora es grabada y almacenada digitalmente, se denomina audio. Este es el término que frecuentemente se emplea en este trabajo al hablar de los datos de entrenamiento y en el desarrollo de la aplicación.

La representación de esta onda en el dominio del tiempo no es fácil de interpretar directamente. En muchas ocasiones resulta imposible localizar sonidos en una forma de onda e identificar a simple vista de qué clase de sonido se trata. Es por ello por lo que las representaciones en el dominio de la frecuencia o las representaciones tiempo-frecuencia cobran mayor importancia. En este capítulo vamos a explicar cómo se calculan estas representaciones y qué utilidad tiene para la clasificación del estado de la abeja reina.

8.2.1. Dominio Frecuencial

Dominio Frecuencial o Dominio de la Frecuencia se refiere al análisis de una señal respecto de sus frecuencias constituyentes [29]. Las señales como el audio están compuestas por una o más componentes de frecuencia. El rango de frecuencias que se podrán representar depende directamente de la Frecuencia de Muestreo (*Fs*) empleada al convertir el audio en una señal discreta durante el proceso de grabación [30]. En concreto, la máxima frecuencia que se podrá representar será de *Fs/2*, según el Teorema de muestreo de Nyquist-Shannon [31].

8.2.2. Conversión Analógico-Digital

El sonido lo podemos obtener grabándolo con un micrófono, que es un tipo de transductor ya que convierte un tipo de energía en otro. Concreta-

mente, un micrófono convierte la energía mecánica, de las ondas de presión sonoras, en energía eléctrica. Una vez realizada esta conversión de tipo de energía, un Conversor Analógico-Digital (ADC) representa cada muestra como un conjunto de bits. Estas muestras pueden ser almacenadas digitalmente en un dispositivo con mucha facilidad y eficiencia. La conversión analógica-digital consta de las siguientes fases:

Filtrado Antialiasing: un Filtro Antialiasing es un tipo de filtro pasa-bajo, es decir, un filtro que elimina las frecuencias superiores a determinada frecuencia de corte, que en este caso particular es *Fs/2*. El propósito de este filtro es eliminar el efecto del Aliasing. El Aliasing es un fenómeno producido durante la etapa de muestreo de una señal que tiene componentes de frecuencia superior a la frecuencia de corte, ocasionando que parte de ellas se replique erróneamente en las bajas frecuencias, distorsionando la señal e imposibilitando recuperar su información [32].

Muestreo: el muestreo consiste básicamente en capturar y almacenar instantáneas de la amplitud de la señal. Cuanto mayor sea la frecuencia de muestreo, más muestras se capturan por segundo, permitiendo almacenar componentes de frecuencia más altas. La *Fs* de un audio usualmente se mide en kilohertz (kHz) [33]. Algunas de las *Fs* estándar para audio son 44,1 kHz, 22,05 kHz, 16 kHz, 8 kHz, aunque la elección de una frecuencia de muestreo depende de la aplicación.

Normalización: en este proceso se aplican diferentes niveles de ganancia a una grabación para que la amplitud se encuentre dentro de un rango mínimo y máximo [34].

8.2.3. Espectrograma

Un espectrograma es una forma de representar la variación de la amplitud y frecuencia de una señal en el tiempo. Surge de calcular el espectro de frecuencias de la señal en ventanas de tiempo [35]. Es útil para visualizar la información de un audio de forma rápida y comparar distintos audios entre sí.

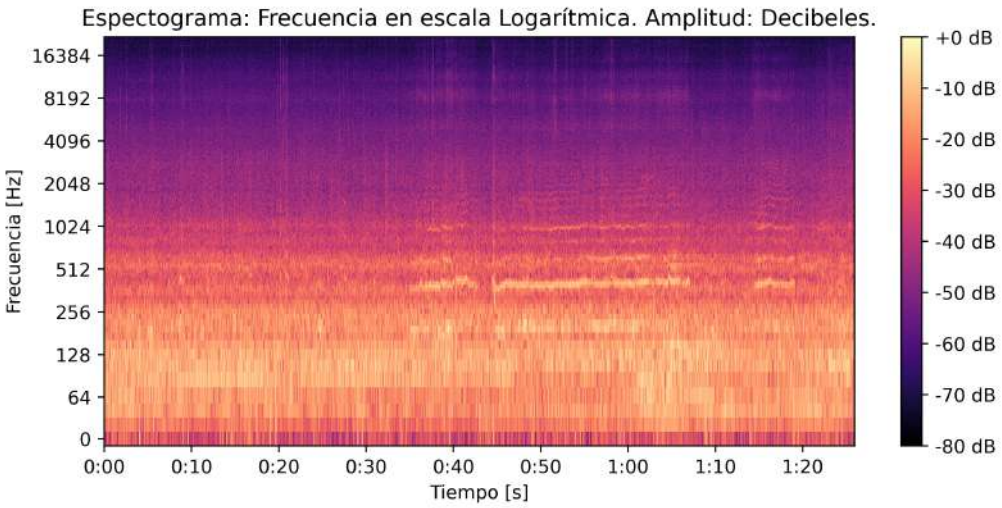

Figura 8.1: Espectrograma de una señal de audio

8.2.4. Extracción de características

8.2.4.1. Escala de Mel

La escala de Mel [36] es una transformación aplicada sobre las frecuencias, pasando de una escala lineal a una escala perceptual. Por ejemplo, un ser humano no percibe igual la diferencia entre 100 Hz y 200 Hz que la diferencia entre 2 kHz y 2,1 kHz, aunque la distancia numérica entre ambos pares es igual.

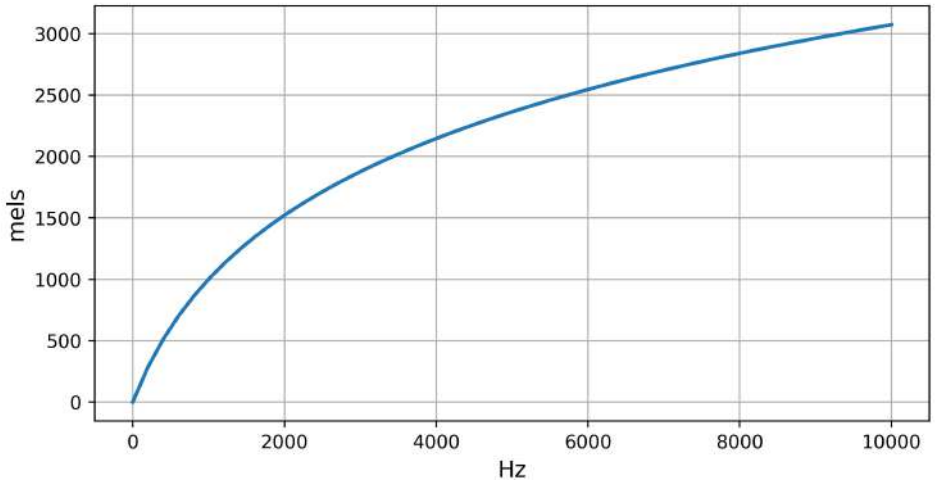

Figura 8.2: Gráfica de conversión Hertz a Mel

8.2.4.2. Coeficientes Cepstrales de las Frecuencias de Mel

Los Coeficientes Cepstrales de las Frecuencias de Mel (MFCC) [37] han sido definidos como coeficientes de representación de una onda de sonido derivados de los coeficientes de la escala de Mel. Para obtener estos coeficientes, el proceso es el siguiente:

1. Segmentar el sonido en tramos de longitud fija.
2. Aplicar a cada segmento la Transformada Discreta de Fourier para separarlo en frecuencias y obtener la potencia espectral (o energía relativa) de cada frecuencia en el segmento.
3. Aplicar la escala de Mel.
4. Tomar el logaritmo neperiano de cada coeficiente de Mel obtenido.
5. Aplicar la transformada de coseno discreta a cada uno de estos logaritmos.

Estos pasos se encuentran en la figura 8.3.

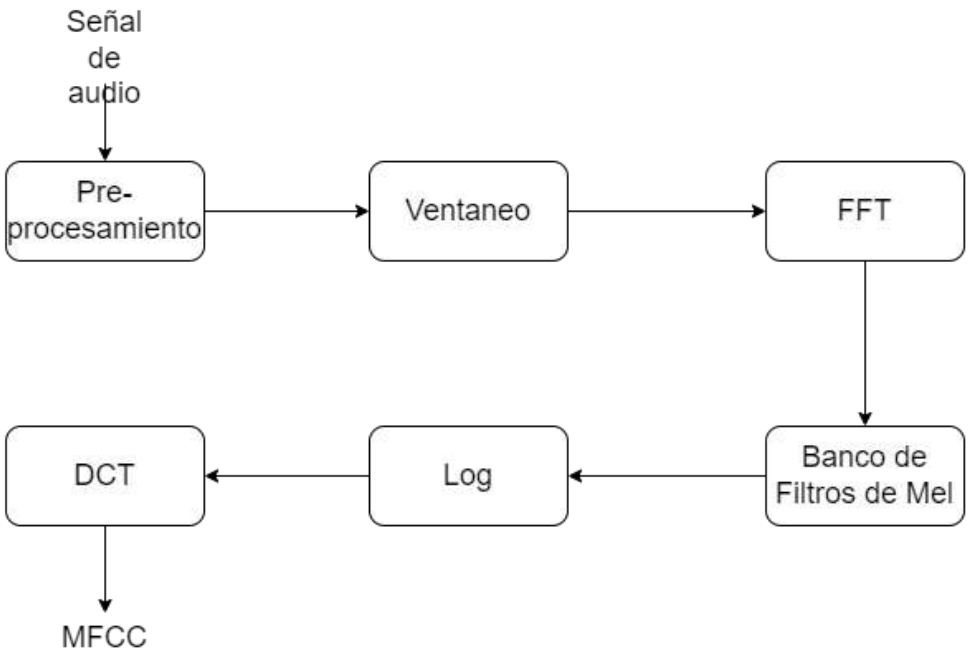

Figura 8.3: Extracción de MFCC

Los MFCC son características (*features*) en forma de vector [37], es decir, se trata de más de un valor, pudiendo obtenerse tantos MFCC como coeficientes de la Transformada Discreta del Coseno (DCT), además de sus derivadas. De cada *frame* se obtiene la Transformada Discreta de Fourier (DFT), la cual se obtiene de la expresión 8.1, obteniendo así la información de cada bloque en el dominio espectral. En la implementación práctica, este proceso es realizado mediante al algoritmo de la Transformada Rápida de Fourier (FFT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}} \quad 0 \leq k \leq N-1 \tag{8.1}$$

Donde *k* es un punto determinado de la DFT de los N posibles, siendo *N* mayor o igual a *L* (longitud de cada bloque de muestras).

Posteriormente, se aplica un banco de filtros de Mel sobre cada *frame* para reducir el espectro en potencia (al cuadrado) a un conjunto de bandas espectrales. A diferencia de la escala lineal de frecuencias utilizada en el cómputo de la FFT, la escala Mel es una escala perceptual de tonalidades equidistantes, es decir, es proporcional al logaritmo de las frecuencias lineales [37]. En la expresión 8.2 se presenta la ecuación que permite pasar de frecuencias a frecuencias Mel.

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{8.2}$$

El banco de filtros de Mel es por tanto un conjunto de filtros triangulares equiespaciados entre sí en la escala Mel, de tal forma que, al volver a la escala lineal de frecuencias, las frecuencias centrales de dichos filtros se distribuyen de forma logarítmica en el espectro, dejando de ser filtros equiespaciados. Para devolver estas frecuencias Mel a frecuencias lineales, basta con aplicar la función inversa de la expresión 8.2, la cual se muestra en la expresión 8.3.

$$f = 700 \cdot \left(10^{\frac{Mel(f)}{2595}} - 1\right) \tag{8.3}$$

En la figura 8.4 se puede observar cómo quedarían estos filtros distribuidos en el espectro. Se trata de un ejemplo en cual se ha utilizado una frecuencia de muestreo de 16 kHz, con un tamaño de FFT de 512 puntos y 6 filtros Mel.

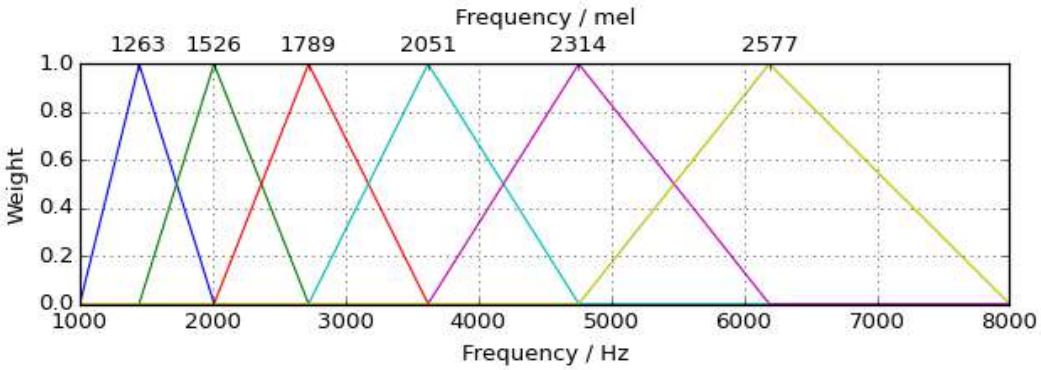

Figura 8.4: Banco de filtro Mel-Frequency [38]

Este conjunto de filtros viene dado según la expresión 8.4, donde *k* es un punto ó *bin* de la FFT, *m* es el número del filtro, siendo *f(m)* el *bin* correspondiente a la frecuencia central del filtro. Por lo tanto, para cada filtro, se tiene un vector de ponderaciones de cada punto de la FFT, ponderando con gran peso aquellos puntos que se encuentren alrededor de la frecuencia central, y con valor nulo para el resto del espectro. En la figura 8.5 se muestra el resultado de cada filtro para cada punto de la FFT. Nótese que los filtros se encuentran perfectamente solapados, de tal manera que el peso total de cada *bin* distribuido en dos filtros es igual a la unidad, satisfaciendo la expresión 8.5 (no necesariamente para el último filtro).

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)}, & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)}, & f(m) \leq k \leq f(m+1) \\ 0, & k > f(m+1) \end{cases} \tag{8.4}$$

$$\sum_{m=0}^{M-1} H_m(k) = 1 \tag{8.5}$$

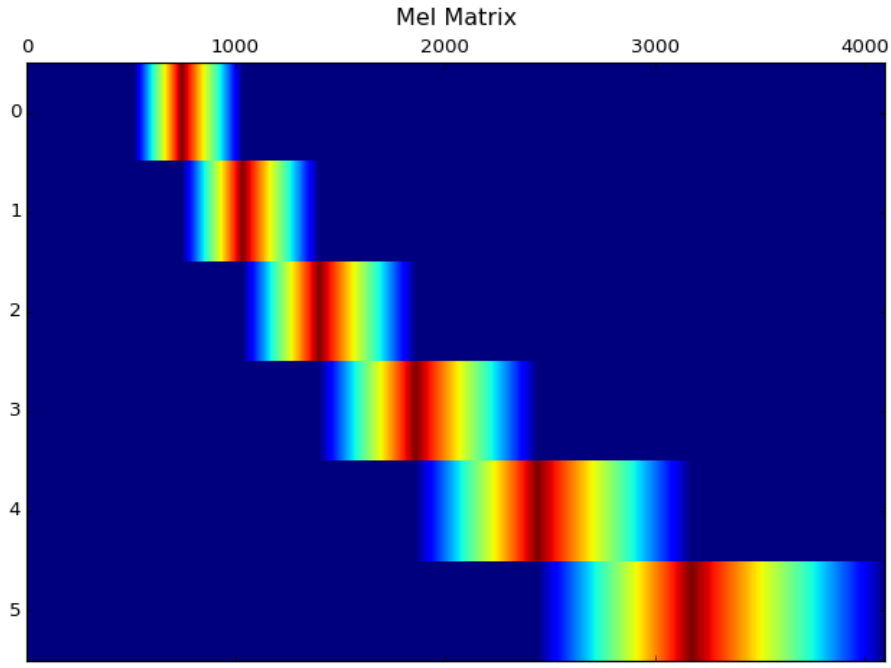

Figura 8.5: Matriz del banco de filtro Mel-Frequency [38]

Este filtrado se aplica a cada *frame*, obteniendo M valores del espectro, correspondientes a cada una de las bandas espectrales definidas por los filtros triangulares. Una vez que se tiene la magnitud de cada una de las bandas, se aplica el logaritmo natural, ya que, la percepción humana de la sonoridad también es procesada de una forma logarítmica. En la expresión 8.6 se muestra la ecuación que describe este proceso [39].

$$S(m) = \log\left[\sum_{k=0}^{N-1} |X(k)|^2 \cdot H_m(k)\right], \quad 0 \leq m \leq M \tag{8.6}$$

Finalmente, se aplica la DCT para obtener el *cepstrum* [1]. En realidad, el dominio cepstral se obtiene mediante la Transformada Discreta de Fourier Inversa (IDFT). Sin embargo, puesto que en el proceso de cálculo de los MFCC no se contempla la parte imaginaria, es habitual realizar este proceso mediante la DCT, la cual se aplica directamente sobre la magnitud logarítmica, según la expresión 8.7.

[1]El término *cepstrum* proviene de las cuatro primeras letras de *spectrum* al revés, ya que, se trata del espectro del logaritmo del espectro de la señal, es decir, trata el logaritmo del espectro de la señal como una nueva señal en sí misma

$$C(k) = \sum_{m=0}^{M-1} S(m) \cos\left(\frac{\pi k(m+1/2)}{M}\right), \quad 0 \leq k < K \tag{8.7}$$

Donde K es el número máximo de coeficientes de la DCT, determinando k el número del MFCC a extraer. M determina el número máximo de puntos sobre los que se aplicará la DCT, que coincide con el número de filtros Mel empleados, siendo m el indicador de cada filtro.

En la figura 8.6 se muestra de forma representativa cada componente de la DCT aplicada sobre un conjunto de 6 filtros. Como se puede apreciar, la primera componente de la DCT *(k=0)* equivale a la energía de la señal, ya que, la ponderación para cada filtro es la unidad. La segunda componente es una relación entre baja y alta frecuencia. A medida que aumenta el orden de las componentes de la DCT aumenta el número de ciclos del coseno. Puesto que se trata de un dominio logarítmico, las sumas y restas equivalen a multiplicaciones y divisiones, respectivamente, por lo que conceptualmente la DCT realiza una serie de ratios entre bandas espectrales. Esto hace que la información más relevante se concentre en los primeros coeficientes, eliminando redundancia.

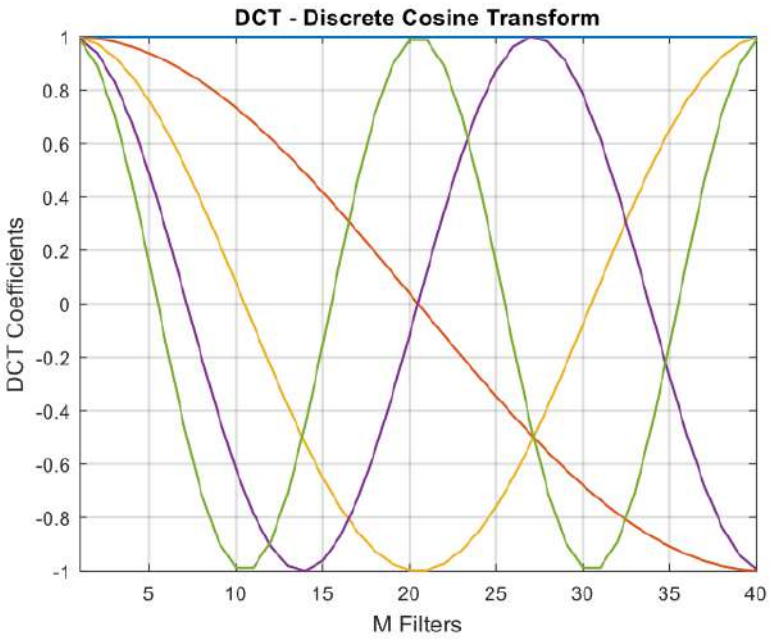

Figura 8.6: Transformada Discreta de Coseno [40]

8.3. Aplicaciones móviles

8.3.1. Definición

Las aplicaciones móviles son programas de software específicamente diseñados para ser ejecutados en teléfonos inteligentes, *tablets* y dispositivos similares. Proveen al usuario de funcionalidades específicas, desde actividades profesionales y acceso a servicios, hasta redes sociales y entretenimiento. Para llevar a cabo sus funciones, las aplicaciones hacen uso de los diversos recursos de hardware disponibles tales como cámaras, acelerómetros, micrófonos, conectividad a internet, Sistema de Posicionamiento Global (GPS), entre otros. Por lo general cuentan con una interfaz gráfica intuitiva y visualmente atractiva. Además de esto son diseñadas para ser ligeras y compatibles con una amplia gama de dispositivos.

8.3.2. Surgimiento y Evolución de las Aplicaciones Móviles

Las aplicaciones surgieron en los años 90, con la llegada de los teléfonos móviles. Por aquel entonces los fabricantes de los sistemas operativos no facilitaban el desarrollo de aplicaciones, pues tenían un estricto control sobre sus sistemas. Sin embargo, esto cambió en 2008 cuando Apple incorporó la App Store a sus productos. En ella el usuario podía descargar aplicaciones de externos. Esto abrió las puertas a muchos desarrolladores para programar aplicaciones sin las restricciones que previamente existían para ello. Más tarde surgió también la Play Store de Google.

8.3.3. El Sistema Operativo Android

Android [41] es un sistema operativo para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, *tablets* y relojes inteligentes. Se trata del sistema operativo móvil más usado a nivel global (superior al 90 % en

el año 2018). Está programado en los lenguajes C, C++, Kotlin, y Java, entre otros [42].

8.3.4. Herramientas de Desarrollo

Para el desarrollo de software en general, existen diversas herramientas que ayudan a ahorrar tiempo y permiten mejorar el escalamiento de los proyectos grandes. El desarrollo de aplicaciones móviles no es una excepción. A continuación se detallan algunos de los conceptos clave relacionados al desarrollo de aplicaciones móviles en Android:

- **Interfaz de Programación de Aplicaciones (API)**: es un conjunto de definiciones y protocolos que permite a dos aplicaciones comunicarse entre sí para compartir información y funcionalidades [43]. Básicamente facilitan que distintos servicios se comuniquen sin necesidad de conocer cómo están implementados internamente.

- **Kit de Desarrollo de Software (SDK)**: es un conjunto de herramientas de desarrollo de software que permite a los programadores crear una aplicación informática para un sistema concreto, por ejemplo ciertos paquetes de software, entornos de trabajo, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etcétera [44]. Usualmente los SDK son proporcionados por fabricantes o empresas. Por ejemplo, el SDK de Android fue desarrollado por Google para permitir y facilitar la implementación de aplicaciones móviles de terceros en dispositivos con sistema operativo Android. El contenido del *kit* puede variar dependiendo del caso. Un SDK básico incluye un depurador, un compilador y varias APIs. También suelen incluir documentación, bibliotecas, controladores y otras herramientas.

- **Kit de Desarrollo Nativo (NDK)**: es un conjunto de herramientas que te permite usar código C y C++ con Android, y proporciona bibliotecas de plataforma que puedes usar para administrar actividades nativas y

acceder a componentes de dispositivos físicos, como sensores y entrada táctil [45]. En definitiva, los NDK son extensiones de los SDK que permiten emplear lenguajes de más bajo nivel que Java para mejorar el rendimiento en tareas con alta carga computacional, además de permitir la reutilización de bibliotecas de C y C++.

- ***Framework***: se trata de un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar [46]. Existen muchos *frameworks*, para diversos lenguajes de programación, cada uno de ellos orientado al desarrollo de algún tipo específico de software, como son las aplicaciones móviles. En este proyecto se emplea el *framework* Kivy [47], para programar aplicaciones móviles en lenguaje Python. De forma adicional, se emplea KivyMD [48] para conseguir una interfaz de usuario estética. KivyMD incorpora una colección de *widgets* de *Material Design* que mejora el aspecto visual de Kivy. Por su parte, *Material Design* [49] es un lenguaje de diseño desarrollado por Google.

- **Empaquetador**: un empaquetador en desarrollo de software es una herramienta que agrupa y organiza archivos, dependencias y configuraciones necesarias de una aplicación en un solo paquete. Su principal función es asegurar que el software pueda ser distribuido y ejecutado de manera uniforme en distintos entornos sin requerir configuraciones adicionales por parte del usuario final. Permite portabilidad, agiliza el despliegue y gestiona de dependencias. En este proyecto se emplea Buildozer [50] como empaquetador.

- **Entorno de Desarrollo Integrado (IDE)**: es un programa que engloba una amplia gama de herramientas para agilizar el desarrollo de software, tales como un editor de código, depuradores, compiladores e intérpretes, gestión de proyectos y archivos, integración con control de versiones, entre otras [51].

- **Git**: es un sistema de control de versiones que realiza un seguimiento de los cambios en los archivos de un proyecto. Además facilita el trabajo colaborativo entre desarrolladores [52].
- **GitHub**: es una plataforma colaborativa online que almacena código fuente y documentación empleando el sistema de versionado con Git [52] [53].

8.4. Abejas Apis Mellífera

La abeja Apis Mellifera, también conocida como abeja europea, es la más común de las especies de abejas en el mundo, y una de las 5 especies que producen miel. Proviene de Europa, África y parte de Asia, y también ha sido implantada en Oceanía y América [54].

8.4.1. Características

Las abejas se dividen en tres castas: obreras, zánganos y la reina. Solo hay una reina en cada colmena, y es la única que puede poner huevos fertilizados. La reina puede llegar a vivir 6 años. Las obreras son abejas hembra que se dedican a los distintos trabajos necesarios para la supervivencia de la colmena: alimentan a las crías, a la reina y a los zánganos, construyen los panales, elaboran miel, protegen la colonia, recolectan recursos fuera de la colmena (actividad llamada pecoreo), realizan tareas relacionadas a la limpieza, entre otras actividades. Una abeja obrera puede vivir entre 6 y 8 semanas en verano (cuando está sometida a mucho trabajo) y entre 4 y 6 meses en invierno. Los zánganos tienen un rol fundamental en la reproducción. Además ayudan a mantener la temperatura y a procesar parte del néctar que han recolectado las obreras. Pueden vivir 3 meses. La reina es la que mantiene la unión de la colmena, debido a la producción de feromona que segrega y va diseminando por toda la colonia. Por ende, cada colmena tiene su olor particular, lo que dificultará el simple hecho de querer incorporar otra madre y que estas puedan ser aceptadas. También es muy difícil que las abejas puedan aceptar una reina nueva sin tener en cuenta su capacidad de puesta de huevos, factor importante para su aprobación. Una colmena puede quedarse sin reina cuando ésta muere o se marcha durante un enjambrazón. También puede pasar que durante el trasiego el apicultor no encuentre a la reina y esta no se halle entre las abejas que fueron puestas en la nueva colmena. Además, cuando la colmena intenta reemplazar naturalmente a la reina es posible que las nuevas candidatas mue-

ran al enfrentarse y luchar entre ellas. Dado que una abeja reina tarda 16 días en nacer, el tiempo es crítico para preservar la colmena.

8.4.2. Procedimiento de Introducción de una Nueva Reina

En caso de que la reina actual sea vieja (mayor a 4 años), el apicultor puede optar por reemplazarla. Existe un procedimiento para introducir una nueva reina, el cual consiste en matar a la reina original y dejarla en el fondo de la colmena. Esta técnica hace más evidente su desaparición, ya que el cadáver está dentro de la colmena, despertando así, el instinto de reemplazo. Pasadas 24 a 48 horas se podrá introducir la caja con la nueva reina. Por último, un paso importante es no volver a revisar hasta pasados de 10 a 12 días de la introducción. Siendo el tiempo indispensable para que esta nueva reina, esté en armonía con el resto de la población y comience su puesta sin ninguna alteración, producto de revisiones inoportunas. El tiempo de espera es crucial, para que la reina no sea apelotonada y eliminada.

8.4.3. El Sonido Producido por las Abejas

Las abejas producen sonido y vibraciones principalmente con sus alas y músculos del tórax. El rango de frecuencias varía dependiendo de la edad de la abeja, la casta (obrera, reina, zángano), y otros factores. Según diversos artículos, la mayor parte de la potencia se encuentra entre los 100 Hz y 2000 Hz [6], y es allí donde se encuentran las componentes más relevantes para la detección del estado de las colmenas.

Parte II

Desarrollo

CAPÍTULO 9

Desarrollo Técnico

9.1. Factibilidad

El proyecto cuenta con los recursos necesarios para su implementación, lo que asegura su viabilidad técnica. Entre estos recursos se incluyen:

Infraestructura y Equipos:

- Disponibilidad de apiarios y herramientas específicas para el manejo de colmenas, como trajes protectores, ahumadores y pinzas universales.
- Equipos tecnológicos adecuados para el desarrollo y pruebas de la aplicación, incluyendo computadoras, dispositivos móviles y micrófonos.

Recursos Humanos y Conocimiento Técnico:

- Integrantes capacitados que aportan parte del conocimiento técnico requerido. Se dispone de experiencia previa en apicultura y desarrollo de modelos de ML, lo cual facilita la implementación y validación del proyecto.

Recursos Tecnológicos y Datos:

- Acceso a conjuntos de datos con audios de colmenas.
- Herramientas tecnológicas para el diseño, desarrollo y pruebas de la solución propuesta.

En el capitulo IV se brindan más detalles sobre el análisis económico-financiero y los montos estimados para el caso de iniciar desde cero un proyecto como este, es decir, considerando el costo de los recursos humanos, materiales, dispositivos y herramientas. En resumen, se evaluó el flujo de caja tanto del proyecto como del inversionista en un periodo de 5 años para estimar el retorno de inversión.

A continuación se presenta el desarrollo técnico que consta de 4 partes. La primera hace referencia al análisis exploratorio de los datos, donde se describen los *datasets* utilizados para el entrenamiento y validación de los modelos de *machine learning*. En la segunda parte se presenta el procedimiento de entrenamiento y validación de los dos modelos utilizados. El primero es el modelo detector, que clasifica si el sonido de un audio corresponde o no a una colmena de abejas Mellifera. El segundo modelo se encarga de hacer la inferencia de uno de los 4 posibles estados de la abeja reina, denominado como modelo multiclase. La tercera parte se centra en el desarrollo de la aplicación y, por último, se cuenta con una parte de trabajo en campo para verificar el funcionamiento de la aplicación móvil.

9.2. Análisis Exploratorio de Datos (EDA)

9.2.1. Datasets

Para el desarrollo del modelo multiclase, se emplearon 2 *datasets* disponibles en línea. El primero de ellos consiste en un conjunto de 7100 audios de colmenas de abejas, de un solo canal y con una duración de 60 segundos cada uno, el *dataset* pesa aproximadamente 21 GB [55]. Los audios de este *dataset* están etiquetados en cuatro clases: reina original, reina no presente, reina rechazada y reina aceptada. El segundo *dataset* consta de 576 audios estéreo etiquetados en dos clases: "reina presente" y "reina no presente". La mayoría de estos audios tiene una duración cercana a los 10 minutos, el *dataset* pesa aproximadamente 51,4 GB [14]. Debido a la diferencia de clases, los audios etiquetados como "reina presente" en el segundo *dataset* se combinaron con los audios de "reina original" del primer *dataset*. Lo mismo se hizo para los audios etiquetados como "reina no presente". Para las clases de "reina rechazada" y "reina aceptada," disponibles solo en el primer *dataset*, se recurrió a una de las técnicas de aumento de datos denominada Synthetic Minority Oversampling Technique (SMOTE) [56].

Para el desarrollo del modelo detector, además de los dos *datasets* anteriores, se utilizaron 8 *datasets* adicionales. Esto se realizó para diversificar los datos y, además, teniendo en cuenta la necesidad de incluir audios que no correspondan a colmenas. Los *datasets* empleados se dividen en dos tipos: los que corresponden a audios de colmenas y los que no. A continuación se describen los 10 *datasets* utilizados en el entrenamiento del modelo detector.

En primer lugar, los *datasets* que contienen audios de colmenas son [55] y [14], los cuales fueron descritos anteriormente, utilizados para el entrenamiento del modelo multiclase y también para el modelo detector. El tercer *dataset* utilizado contiene 248 audios de colmenas de 30 segundos cada uno [57]. Estos audios fueron capturados por un sistema denominado Inspector Visual de Abejas. Cabe destacar que este *dataset* incluye imágenes, información de

temperatura, humedad, presión entre otros; el directorio que contiene los audios es *SoundLog*.

Con la idea de realizar un trabajo de campo complementario y brindar robustez al modelo detector, se tomaron muestras propias para reforzar el sonido de las colmenas. Se adquirieron 16 grabaciones del sonido de 10 colmenas en la zona de 25 de Mayo, Misiones, Argentina. De estos audios, se utilizaron 10 para el entrenamiento del modelo detector, mientras que los restantes fueron destinados para testear el modelo y la aplicación móvil. Los audios no tienen una longitud fija, son audios de entre 40 minutos hasta 7 horas. El tamaño del *dataset* es de 1,84 GB. Estos datos cobran una mayor importancia, teniendo en cuenta que es el entorno en el cual funcionará el modelo.

El quinto conjunto de datos utilizado es *Bee Audio Object Detection*, el cual contiene 2840 audios donde el 50 % corresponden a sonido de colmenas y la otra mitad del *dataset* no corresponde a sonido de colmenas, son audios de 10 segundos [58].

Así como el *dataset* anterior donde ya incluye sonidos que no corresponden a colmenas, se emplearon otros *dataset* con audios que no corresponden a sonido de colmena. Uno de ellos es *ESC-50*, contiene 2000 audios con sonidos de 5 categorías, como ser sonidos de animales, naturaleza, humanos no verbales, del hogar y de exteriores. Dentro de cada categoría, estos sonidos presenta 50 clases y cada audio tienen una duración de 5 segundos [59].

También se cuenta con un séptimo *dataset* que presenta audios de la Selva Misionera, el cual incluye 166 grabaciones de aproximadamente 10 minutos cada una. Además, se empleó el *dataset Traffic*, que contiene 200 audios de sonidos de automóviles, cada uno con una duración de 3 segundos [60]. Este último es importante considerado que algunos apiarios se sitúan cerca de zonas transitadas por vehículos.

El noveno *dataset* utilizado es *urbansound8k*, que contiene 8732 audios clasificados en 10 tipos de sonidos, como aire acondicionado, bocina de auto, niños jugando, ladrido de perro, taladro, motor en marcha, disparo, martillo

neumático, sirena y música callejera [61]. Estos audios están en formato estéreo y tienen una duración menor a 4 segundos.

Con el objetivo de complementar los audios con sonidos que no son de colmenas, se utilizaron los audios de videos de *YouTube*, conocidos como *youtube-8m* [62]. Este es un caso particular, ya que no existe un *dataset* estructurado como tal, sino que consiste en un archivo que contiene los nombres y las *URLs* de los videos [63]. Para construir el *dataset* a partir de los audios de estos videos, se implementó un *script* utilizando la biblioteca *yt-dlp* para acceder a los videos y extraer el audio [64]. Aunque la mayoría de la documentación recomienda usar la biblioteca *youtube-dl*, esta no está disponible. Además, es importante considerar que algunos videos pueden no estar accesibles en la plataforma. En cuanto a la obtención de los audios, también se debe tener en cuenta el formato en el que se guardarán; en este caso, los audios se almacenaron en formato *mp3*. Se lograron obtener 664 audios con una duración variable de entre 3 y 60 minutos.

A continuación se presenta la tabla 9.1 en la que se resumen los *datasets* utilizados y las características principales de cada uno.

Tabla 9.1: Resumen de los *dataset* y sus características

Nombre	Cantidad	Sonido de Colmenas	Tamaño	Duración de muestra
BeehiveSounds [55]	7100	Sí	21 GB	60 segundos
Nolasco2019 [14]	576	Sí	51.4 GB	10 minutos
BUT-2 [57]	248	Sí	675 MB	30 segundos
Apiario en 25 de Mayo (Campo)	10	Sí	1.84 GB	40 minutos a 7 horas
Bee Audio Detection [58]	2840	Compuesto	4,66 GB	10 segundos
ESC-50 [59]	2000	No	843 MB	5 segundos
Selva Misionera (FIO)	166	No	387 MB	10 minutos
Traffic [60]	200	No	101 MB	3 segundos
urbansound8k [61]	8732	No	6,61 GB	<4 segundos
youtube-8m [62]	664	No	3,52 GB	3 a 60 minutos

9.2.2. Actividades de procesamiento

Para el procesamiento de los datos se utilizaron dos *frameworks*: PyCaret [65] y CleanLab [66], [67]. PyCaret es una librería de código abierto que automatiza el flujo de trabajo con modelos de ML, mientras que CleanLab facilita la tarea de corrección de etiquetas.

En el procesamiento de los datos utilizados, en primer lugar se realizó un EDA de los audios para el modelo clasificador multiclase [68]. Se calcularon los espectrogramas de cada uno de los audios para el análisis visual. Esto permitió encontrar defectos en algunos de los segmentos de audio del primer *dataset*. En concreto, todos los segmentos número 5 presentaban algún tipo de distorsión, por lo cual fueron descartados. Se desconoce la causa que originó estos problemas. Para más detalles remítase al artículo de preprocesamiento [68]. Posteriormente, utilizando Cleanlab, se corrigieron errores de etiquetado mediante una técnica de aprendizaje no supervisado llamada *clustering*. A continuación, se procedió a la extracción de características de los audios.

9.2.2.1. Obtención de MFCCs

Inicialmente se empleó la librería Librosa para la obtención de los MFCCs. Sin embargo, al momento de implementar el modelo en la aplicación, surgieron algunas dificultades debido a las numerosas dependencias de esta librería y la tediosa compilación de las mismas. Por lo tanto, se optó por desarrollar funciones propias para la extracción de los MFCCs.

El algoritmo planteado consiste en realizar un recorte de los audios en fragmentos de 10 s. Se emplea un filtro pasa bajo, el cual se emplea para eliminar la información presente en el audio por encima de los 2 kHz. El siguiente paso consiste en hacer un re-muestreo para los recortes de audios usando una frecuencia de muestreo de 4 kHz, ya que la información de interés del sonido de una colmena se encuentra en 20 y 2000 Hz. Luego se aplica una normalización entre -1 y 1 de los mismos. Debido a que en algunos segmentos aparecen picos de ruido impulsivo (ocasionados por golpes o zumbidos de abejas muy cerca del micrófono), se implementó un algoritmo para atenuar estos picos teniendo en cuenta la media y la desviación estándar.

Una vez realizada la adecuación anterior del segmento, se procede al cálculo de los MFCCs del mismo. El algoritmo para el cálculo de los MFCCs

contempla los pasos descritos en la sección 8, donde al segmento de audio se realiza un ventaneo de 2048 muestras, con una superposición entre ellos de 508 muestras. Luego se emplea la ventana de Hanning, la cual se utiliza para evitar un corte abrupto y altas frecuencias. A partir de esto, se procede a obtener la FFT de cada segmento y aplicar el banco de 128 filtros de Mel. A continuación, se realiza una conversión de estos valores a la escala logarítmica, para luego aplicar la DCT tipo 2 y así obtener los MFCCs.

Una vez obtenidos los 50 MFCCs, fueron guardados en archivos *csv* donde fueron etiquetados como “0” en caso de sonido de colmena y “1” el sonido de otra clase, esta es la metodología tomada para el modelo detector.

Para el modelo multiclase, las etiquetas correspondiente son “0” reina presente, “1” reina ausente, “2” reina presente rechazada, “3” reina presente aceptada.

9.3. Modelos

En la sección anterior se destacó el procesamiento de los datos de entrada para el entrenamiento de los modelos de aprendizaje automático. La cantidad y diversidad de estos datos influyen directamente a la generalización y la precisión de los modelos. En esta sección se describe el proceso de entrenamiento y adaptación de los modelos para su posterior integración en la aplicación móvil. Además, se detallan las tecnologías y herramientas empleadas.

9.3.1. Tecnologías y Herramientas

Lenguaje de programación: el lenguaje utilizado para el desarrollo de modelos es Python. Este lenguaje de propósito general es muy popular en el campo de la IA. Tiene un amplio abanico de librerías para todo tipo de aplicaciones, entre las cuales se destaca el aprendizaje automático.

Frameworks: se emplearon dos *frameworks* para la etapa de optimización del *pipeline* de procesamiento: PyCaret y CleanLab.

- PyCaret: *framework* para automatizar los flujos de trabajo de aprendizaje automático. Se eligió este framework por la facilidad de uso que presenta, la cantidad de modelos que implementa y la versatilidad para probar distintas configuraciones.
- CleanLab: *framework* para la corrección automática de etiquetas. Se trata de una librería muy potente y moderna para mejorar la calidad de los datos de entrenamiento.

Librerías: las principales librerías empleadas se destacan a continuación:

- NumPy: para operaciones numéricas y matriciales.

- Pandas: utilizada en manejo y análisis de datos.
- Sklearn: empleada en gráficos de matriz de confusión y comparación de métricas.
- Os: usado en la manipulación de nombre de archivos.
- Tensor Flow Lite: para trabajar con modelos de red neuronal.
- SoundFile: para leer audio.

9.3.2. Modelo Detector

Este trabajo se enfoca en la clasificación multiclase del estado de la abeja reina como enfoque general. Al ingresar un audio, el algoritmo procesará y obtendrá los MFCC para luego enviarlos al modelo de ML. En este contexto, el modelo clasificador multiclase por defecto indicará una clase: reina original, ausente, aceptada o rechazada, dado que el mismo no tiene una quinta clase para indicar en caso de que el audio ingresado no corresponda a una colmena. Si bien se puede incorporar fácilmente una clase extra al modelo para cubrir cualquier otro escenario, esto implica que dicho modelo sea más complejo y realice dos tareas al mismo tiempo: clasificar audios que son de abejas y descartar aquellos que no lo son. Es por ello que se optó por otra solución: el desarrollo de un segundo modelo, denominado en este informe como modelo detector, el cual realiza una clasificación preliminar para determinar si el sonido del audio que se quiere utilizar es o no de una colmena de abejas. De este modo se divide la solución del problema en dos etapas, una para cada modelo. Al separar las tareas de detección y clasificación, cada modelo puede optimizarse de manera independiente, reduciendo la complejidad computacional y mejorando la precisión específica de cada etapa. La capacidad de entrenamiento independiente constituye uno de los beneficios más relevantes de esta arquitectura. Desde una perspectiva de mantenimiento y actualización tecnológica, la modularidad facilita intervenciones precisas sin necesidad de reentrenar todo el sistema. Por ejemplo, si se requiere mejorar la capacidad

de detección de sonidos de colmena, es posible actualizar el modelo detector sin impactar la arquitectura del clasificador. Esta flexibilidad permite una evolución incremental del sistema, optimizando recursos computacionales y reduciendo los tiempos de desarrollo. Adicionalmente, la especialización de cada modelo permite implementar estrategias de regularización y ajuste más específicas, lo que potencialmente mejora la capacidad de generalización y reduce el riesgo de sobreajuste en cada etapa del procesamiento.

A continuación se procede a indicar las consideraciones tomadas para el entrenamiento del modelo detector. En cuanto a los datos para el entrenamiento, de un conjunto de 135.756 vectores de coeficientes (cada uno de ellos con 50 MFCCs), el 80 % se destinó para entrenamiento y el 20 % para validación. Cabe aclarar que se cuenta con un conjunto de 15.410 vectores de coeficientes para el testeo del modelo.

Para la creación de este modelo de ML, inicialmente se definió una topología manual usando la librería de Tensor Flow, donde la capa de entrada espera los 50 MFCC obtenidos del audio; luego 3 capas ocultas con 256, 128, 64 neuronas cada una respectivamente y la función de activación implementada es *relu*. Además, se agregaron capas *dropout* para activar y desactivar un 30 % de las neuronas de cada capa. En cuanto a la capa de salida, se definió una capa con dos neuronas para la clasificación, si los MFCC ingresados corresponde o no a un audio con sonido de colmena. La función de activación para esta última capa es *softmax*, la misma cuenta con una ventaja que asigna una probabilidad a cada clase y la suma de probabilidades es uno. Dada la asignación de probabilidad a cada una de las clases, si se quiere saber cual es la clase predicha, tomando el argumento máximo, ya se tiene esta información. El optimizador implementado es la estimación del momento adaptativo *Adam*, el cual está basado en el descenso del gradiente. Este optimizador está pensado para trabajar con grandes cantidades de datos. La función de pérdida usada es la de entropía cruzada *sparse categorical crossentropy* ya que se tienen 2 clases, esta función produce un índice de la categoría con mayor probabilidad de coincidencia. La métrica utilizada es la *accuracy*, ya que mide el porcentaje de predicciones correctas sobre el total de muestras y las clases están balancea-

das. Para evitar un sobre-ajuste, se implementó un *callback* de parada temprana donde la métrica empleada para el monitoreo es la pérdida de la validación *(val loss)*, definiendo una parada temprana para que luego de dos épocas en que las curvas de perdida del entrenamiento y la validación comiencen a separarse.

Con la configuración anterior se procedió a entrenar el modelo de ML durante 19 épocas según la configuración de la parada temprana. En la figura 9.1 se tiene la gráfica del error de entrenamiento y validación, en la figura 9.2 se tiene la gráfica de la precisión de entrenamiento y validación.

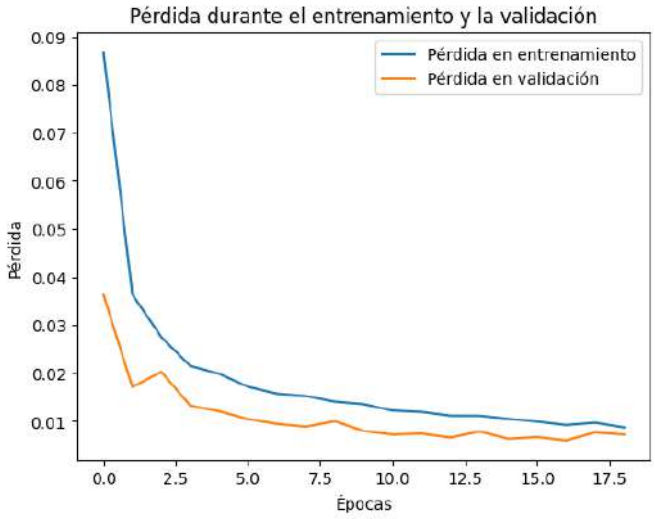

Figura 9.1: Error de entrenamiento y validación

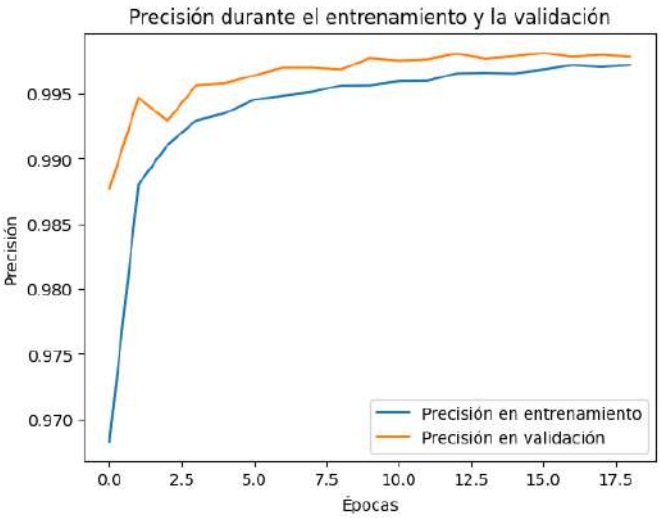

Figura 9.2: precisión de entrenamiento y validación

Además de la topología anterior, se habían probado previamente algunas configuraciones sobre la misma, con la idea de llegar a la topología que

mejor se adapte a los MFCCs. La solución planteada para hallar la topología ideal o que mejor se aproxime de tal forma que quede justificada la elección de la misma, fue el uso de Algoritmos Genéticos (AG), los cuales realizan variaciones y distintas pruebas para los parámetros que se desean modificar (genes), encontrando la combinación óptima para el entrenamiento del modelo. A continuación se explica el procedimiento y las consideraciones tomadas para la topología lograda mediante AG [69], [70].

En lo que respecta a la implementación de AG, los genes a los que se decidió utilizar para obtener el número óptimo fueron la cantidad de capas densas, el número de neuronas en cada capa y el *dropout*. Por lo tanto, los 3 genes mencionados fueron empleados para determinar el número óptimo del cromosoma. La cantidad de generaciones empleadas fue de 10, con una población de 10 integrantes y una tasa de mutación del 5 %. Se entrenó la red asociada a cada individuo de la población durante 3 épocas, para la comparativa de *best fitness*. La elección de esta cantidad de épocas se debe, por un lado, a la reducción del tiempo de ejecución, y por otro lado, para priorizar la calidad de los hiperparámetros por sobre el tiempo de entrenamiento. Más adelante, una vez fijados los hiperparámetros, se aumentó la cantidad de épocas.

La topología lograda fue de 3 capas ocultas, con 148, 105, 374 neuronas cada capa respectivamente, un *dropout* del 70 %. Los demás parámetros como la función de activación, capa de entrada y salida, optimizador y división de los datos se mantuvo igual que el caso de la topología definida manualmente.

Una vez obtenidos los valores anteriores para cada uno de los genes se procedió a un entrenamiento del modelo, donde se implementó parada temprana, logrando que el modelo se entrene durante 21 épocas. En la figura 9.3 se tiene la gráfica del error de entrenamiento y validación, en la figura 9.4 se tiene la gráfica de la precisión de entrenamiento y validación.

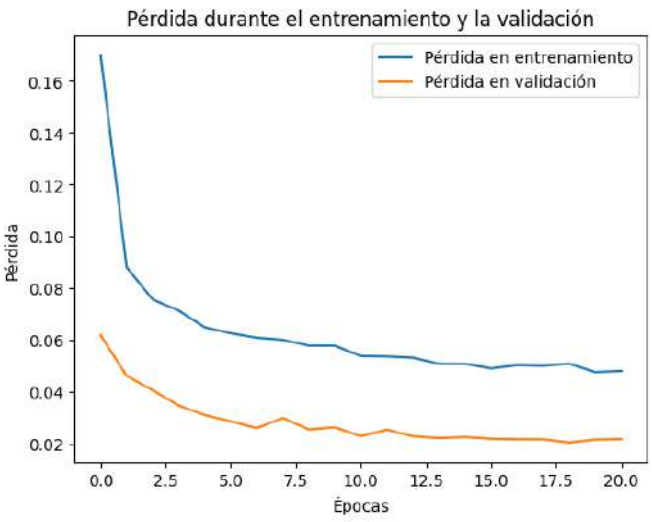

Figura 9.3: Error de entrenamiento y validación

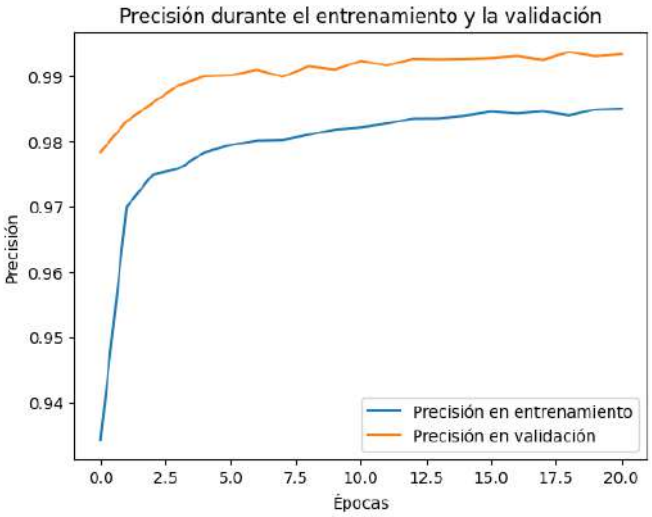

Figura 9.4: Error de entrenamiento y validación

A continuación se presenta en la tabla 9.2 una comparativa entre estos dos modelos. Si bien las métricas de desempeño de la topología manual presentan valores ligeramente superiores, se elige la topología de AG fundamentada en la automatización de la selección de parámetros e hiperparámetros, que permite una optimización más eficiente y menos dependiente. Para el caso que se realice alguna implementación de más datos o algún otro cambio en el conjunto de entrenamiento, sería necesario volver a encontrar los parámetros óptimos, por lo tanto el uso de AG siempre sería el camino recomendado, dado que hacer ajuste manual y encontrar la topología que mejor se ajusta toma un tiempo que no se justifica.

Tabla 9.2: Comparativa entre topologías modelo detector

Modelo	Precisión Train	Precisión Val.	Precisión Test	Tamaño
Topología Manual	99,71 %	99,78 %	99,78 %	110 KB
Topología de AG	98,53 %	99,33 %	99,37 %	128 KB

9.3.3. Modelo Multiclase

El modelo multiclase es el más importante en este trabajo, por lo tanto se probaron optimizaciones, mejoras de topologías e implementación de AG para la obtención de los parámetros óptimos de la topología.

Como primer modelo multiclase, se obtuvo mediante el *framework* PyCaret, un modelo con 84,69 % de precisión en el entrenamiento, y un 91,5 % en el de test (figura 9.5). Este *framework* permitió optimizar el *pipeline* de procesamiento previo al modelo. Luego de tener el *pipeline* optimizado se pasó a implementar una Red Neuronal Artificial Convencional (RNA), dado que es más práctica de implementar en Android.

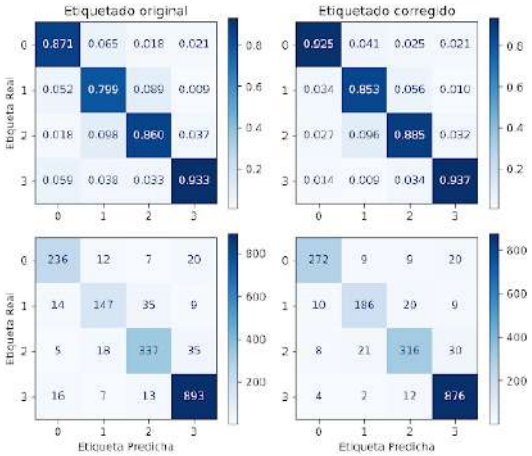

Figura 9.5: Matriz de confusión del primer modelo

Planteando la topología manual usando la librería de Tensor Flow, con 3 capas ocultas de 256, 128 y 64 neuronas respectivamente, con la función de activación de *relu* cada una. A la salida se tiene una función de *softmax* dadas las ventajas descritas anteriormente.

En cuanto a las demás consideración de los parámetros de entrenamiento se tomaron las mismas que para el modelo detector; en la figura 9.6

se tiene el error de entrenamiento y validación, y en la figura 9.7 se tiene la precisión de 95,75 % para el entrenamiento y 97,33 % para el test.

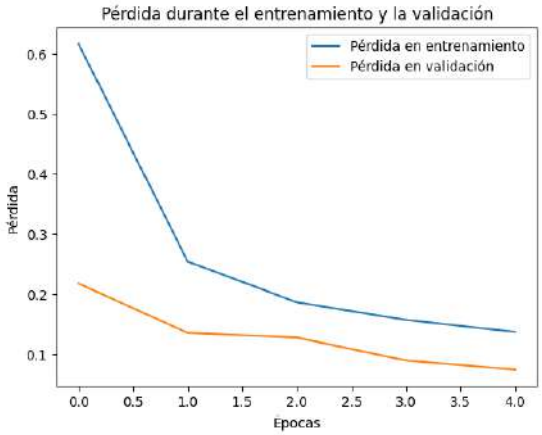

Figura 9.6: Error de la topología manual

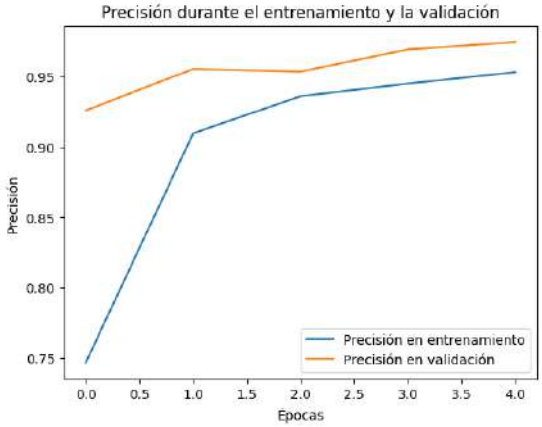

Figura 9.7: Precisión modelo topología manual

La topología resultante del AG, fueron con 3 capas ocultas de 650, 750 y 800 neuronas respectivamente. Se obtuvo una precisión del 98,36 % para el entrenamiento tal como se ve en la figura 9.8 y 98,55 % para el test.

En la tabla 9.3 se muestra de forma resumida la comparativa entre los valores de entrenamiento y testeo del modelo de ML. Además, se muestra el tamaño de los mismos. La topología adoptada para el modelo multiclase es la lograda mediante AG, donde además de las ventajas descritas en la sección 9.3.2, en este caso logró el mejor rendimiento en el entrenamiento y testeo.

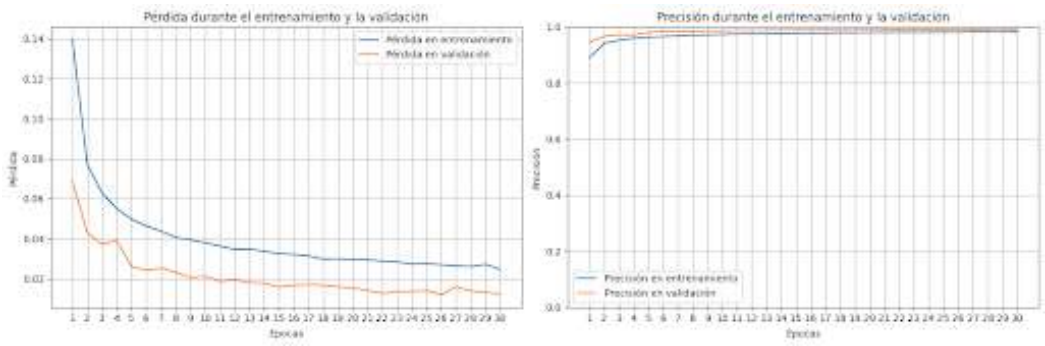

Figura 9.8: Error y precisión de entrenamiento

Tabla 9.3: Comparativa entre topologías modelo multiclase

Modelo	Precisión Train	Precisión Test	Tamaño
Extra Trees Classifier	84,69 %	91,5 %	142 MB
Topología Manual	95,75 %	97,95 %	110 KB
Topología de AG	99,08 %	98,06 %	2,2 MB

9.3.4. Compresión de Modelos

Como los modelos anteriores deben ser implementados en una aplicación móvil, se deben tener en cuenta el formato y tamaño de los mismos. En primer lugar se tiene la conversión de formato. Los modelos se pueden guardar en formato *keras*, pero en este caso, para la implementación de los mismos es necesario pasar a formato *tflite*. La librería Tensor Flow Lite, está pensada para implementar modelos en este formato para Android, y sus dependencias se incorporan fácilmente al proyecto durante la compilación de la aplicación. En cuanto a los procesos de compresión y optimización de modelos, hay varias técnicas, a continuación se presentan las técnicas experimentadas.

En primer lugar se implementó la conversión tradicional sugerida en la documentación de Tensor Flow, la cual implica en una optimización del modelo por defecto y luego una cuantización; por defecto los números que se usan para representar la precisión son en formato de punto flotante de 32 bits, por lo tanto se hace una cuantización pasando los pesos a flotante de 16 bits, de esta forma se recorta la cantidad de decimales, lo que permite reducir el tamaño del modelo y mejora el procesamiento. Una vez hecho este recorte se realizó la conversión y guardado en formato *tflite*.

Otra implementación de compresión es la optimización previa al entrenamiento, esta consiste en indicar a la topología del modelo que el mismo va a ser cuantizado, esta técnica resulta en un 75 % de optimización del modelo, luego se hace el entrenamiento, se cuantiza a 16 bits flotantes para guardar en formato *tflite*; en este caso no se implementó parada temprana ya que al indicar que el modelo va ser cuantizado y convertido mediante la función *tensorflow model optimization* de Tensor Flow, la misma función realiza ajustes con el objetivo de optimizar el tamaño del modelo y los pesos del mismo.

Otra optimización probada fue la poda (*pruning*). Consiste en quitar aquellos parámetros del modelo que tienen un impacto menor en sus predicciones. Los modelos reducidos tienen el mismo tamaño en el disco y el mismo entorno de ejecución, pero pueden comprimirse de manera más eficaz.

Por último se probó el agrupamiento de *clúster* de los pesos en cada una de las capas, esto se hace en un número predefinido de *clúster*, la cantidad configurado fue de 4 *clúster*.

A continuación, se presenta la tabla 9.4 donde se presenta la comparativa del tamaño de los modelos luego de la compresión. A modo de referencia, un modelo en formato *keras* (sin optimización) pesa 1.059 KB.

Tabla 9.4: Comparativa de compresión

Compresión	**Formato**	**Tamaño**
Pre-entrenamiento	tflite	68 KB
Post-entrenamiento	tflite	128 KB
Pruning	tflite	256 KB
Agrupamiento de *Clúster*	tflite	746 KB

9.4. Aplicación

El desarrollo de la aplicación estuvo estrechamente ligado al procesamiento y entrenamiento de los modelos de machine learning. Esto se debe a que la aplicación internamente debe procesar los audios de entrada de la misma manera en que fueron procesados los audios usados en el entrenamiento. Por ello varias de las librerías son las mismas que las descritas en la sección 9.3.1.

9.4.1. Tecnologías y herramientas Empleadas

Lenguaje: la aplicación fue desarrollada en el lenguaje Python. Para permitir la ejecución en Android se empleó la herramienta de desarrollo Python for Android (p4a). Emplear Python en el desarrollo de la aplicación permite una integración más fácil de los modelos desarrollados previamente en el mismo lenguaje.

Framework: como se mencionó brevemente con anterioridad, para desarrollar la aplicación se empleó el *framework* KivyMD, que puede considerarse una extensión de Kivy que mejora el aspecto visual de los componentes de la Interfaz Gráfica de Usuario (GUI).

Librerías:

1. NumPy: para operaciones numéricas y matriciales.
2. Tensor Flow Lite: para trabajar con modelos de red neuronal.
3. SoundFile: para cargar audio desde memoria en formato WAV.
4. PyJNIus: para emplear clases de Java en Python, mediante la Interfaz Nativa de Java (JNI).

Empaquetador: el empaquetador Buildozer fue escogido para compilar el código y demás recursos necesarios en formato de Paquetes de Aplicaciones para Android (APK), dado que es muy frecuente su uso con Kivy. Buildozer integra diversas tecnologías para llevar a cabo su tarea. La configuración del mismo se lleva a cabo mediante el archivo *buildozer.spec*, que se halla en el mismo directorio que el código fuente. Contiene una amplia variedad de parámetros que pueden ajustarse, incluyendo las dependencias adicionales requeridas, las arquitecturas de hardware para las cuales compilar la aplicación, recursos estáticos que deben incluirse en el APK, entre muchos otros. Este archivo se genera automáticamente por Buildozer con un comando de inicialización, y viene preconfigurado y comentado de modo que el desarrollador cuente con una plantilla para comenzar a trabajar. Muchos de los parámetros debieron configurarse para adecuar la compilación a las necesidades de este proyecto.

En cuanto a la arquitectura de hardware objetivo, se realizó una investigación para conocer estimaciones sobre el porcentaje de dispositivos existentes (cuota de mercado) para cada una de las arquitecturas disponibles para *smartphones*. En la tabla 9.5 se resume la información recabada de diversas fuentes [71], [72], [73], [74]. En resumen, la arquitectura de 64 bits arm64-v8a es la predominante en la actualidad, por lo cual fue escogida para la compilación del APK final. Previamente también se realizaron pruebas exitosas compilando para la arquitectura armeabi-v7a. Si bien el despliegue no forma parte del presente desarrollo, es interesante conocer este aspecto a la hora de la compilación, principalmente para reducir el tiempo de dicho proceso. Además, aunque Buildozer permite compilar un solo paquete con soporte para varias arquitecturas simultáneamente, esto aumenta el tamaño del APK.

Tabla 9.5: Arquitecturas de hardware para smartphones y sus respectivas cuotas de mercado

Arquitectura	**Ancho de palabra (bits)**	**Cuota de mercado**
armeabi-v7a	32	4 %
arm64-v8a	64	79 %
ARMv9	64	16 %
x86	32 y 64	<1 %

Control de Versiones: Para garantizar una gestión ordenada y eficiente del desarrollo, se utilizó Git como sistema de control de versiones y GitHub como plataforma de colaboración en línea. Estas herramientas permitieron mantener un historial detallado de los cambios realizados en el código fuente, facilitando el trabajo en equipo y asegurando la trazabilidad de cada modificación.

El flujo de trabajo se estructuró mediante el uso de ramas específicas, las cuales se destinaron al desarrollo de nuevas funcionalidades, corrección de errores y pruebas experimentales. Esto permitió incorporar las características de forma gradual y mantener la estabilidad del proyecto en todo momento.

9.4.2. Actividades de Desarrollo

El proceso de desarrollo se llevó a cabo a lo largo de los meses de Julio hasta Noviembre. A medida que se incorporaron las funcionalidades y módulos, con frecuencia debieron ajustarse uno o más parámetros del archivo de configuración *buildozer.spec*. En primera instancia se eliminaron del archivo aquellos parámetros necesarios para construir el paquete para iOS, dado que esta aplicación está pensada únicamente para Android. Además, para ahorrar tiempo de compilación y evitar copiar archivos innecesarios se excluyeron aquellos directorios relacionados al entorno virtual y caché de Python. También se excluyeron archivos relacionados al repositorio de Git, requerimientos y el propio archivo *buildozer.spec*, para que éste no forme parte del APK. Con esto se consiguió optimizar el proceso de compilación de la aplicación.

A continuación se describen las actividades de desarrollo, divididas en seis etapas: desarrollo de pantallas, lógica de carga de audio, integración de modelos, mejora de la pantalla de muestra de resultados, lógica de grabación de audio, y mejora del aspecto visual. Estas etapas se pueden visualizar en el Diagrama de Gannt de la figura 9.9.

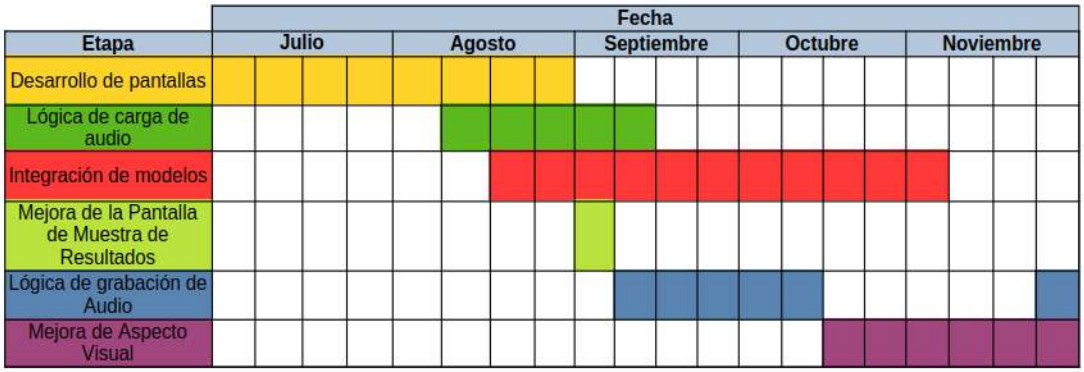

Figura 9.9: Diagrama de Gantt de las etapas de desarrollo de la aplicación

Para brindar más detalles, se describen las actividades realizadas en cada una de las etapas previamente mencionadas.

9.4.2.1. Desarrollo de Pantallas

En una primera etapa se desarrolló la GUI: pantallas, botones, y etiquetas de texto. A esto se sumó la lógica relacionada a la transición de pantallas. En total se han desarrollado cinco pantallas: la pantalla principal, de grabación, de selección de audio, de inferencia, y de muestra de resultados. A lo largo de las siguientes etapas se continuó perfeccionando la GUI, lo cual incluyó aumentar el tamaño de fuente en los botones y etiquetas para facilitar el uso de la app.

A continuación, se muestran las 8 pantallas resultantes de la versión final, para el caso de tema claro.

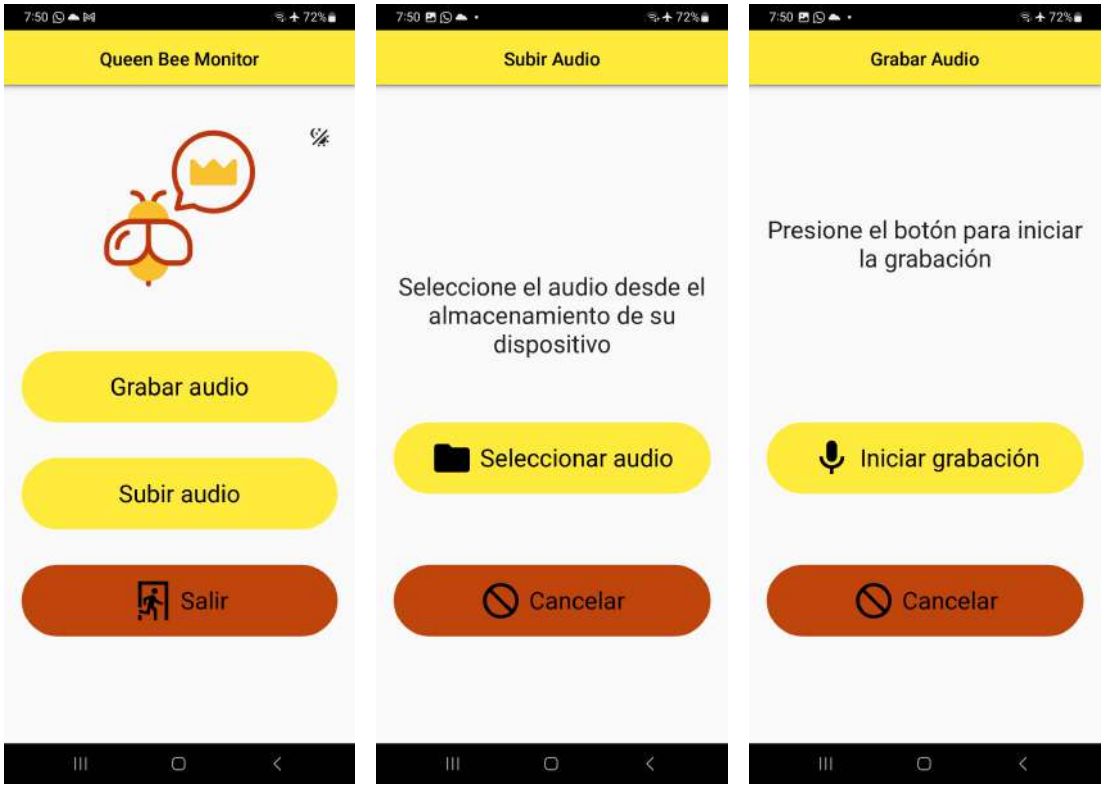

Figura 9.10: De izquierda a derecha: pantalla principal, pantalla de selección de audio, pantalla de grabación.

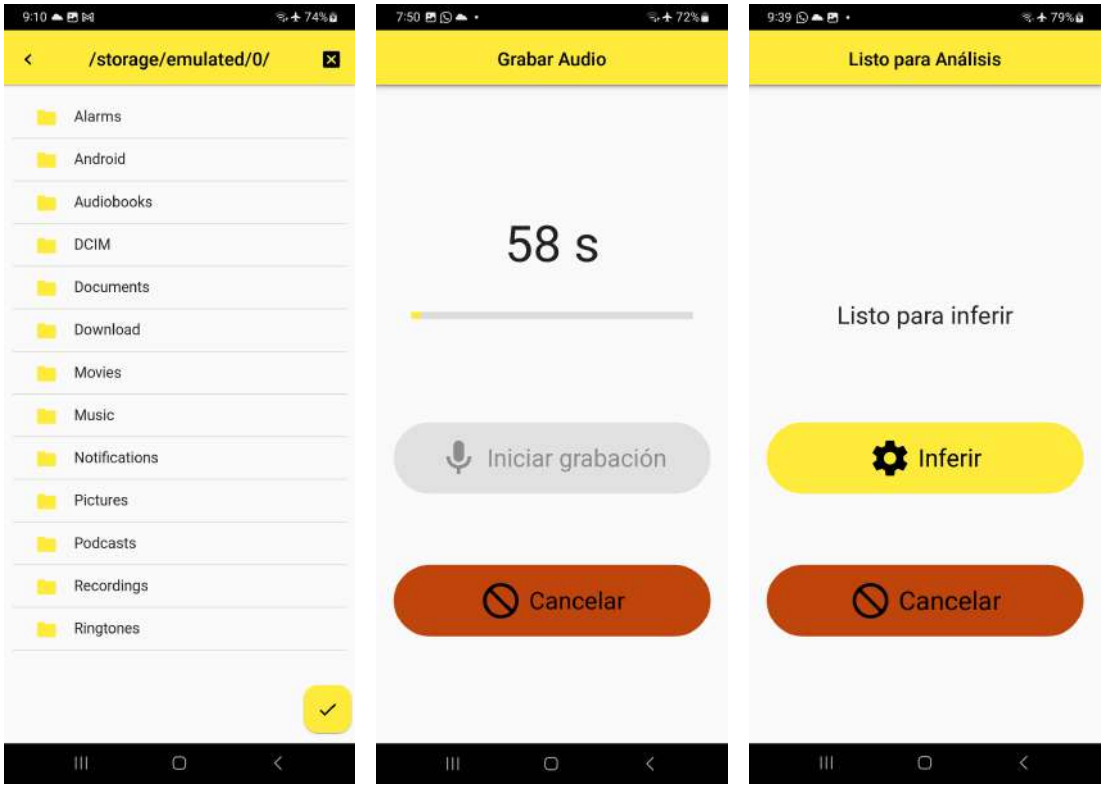

Figura 9.11: De izquierda a derecha: selector de archivos (opción subir audio), cuenta regresiva de grabación (opción grabar audio) y pantalla de inferencia

Figura 9.12: Izquierda: pantalla de estado de abeja reina. Derecha: pantalla de aviso

9.4.2.2. Lógica de Carga de Audio

La siguiente etapa consistió en agregar funcionalidad para el botón que permite seleccionar un audio desde el almacenamiento del dispositivo y emplearlo en la aplicación. Esto requirió incorporar dos permisos en el archivo de configuración de Buildozer: `READ_MEDIA_AUDIO` y `READ_EXTERNAL_STORAGE`. El primero de estos solo es necesario a partir de la versión 13 de Android. De este modo el Sistema Operativo es avisado de los permisos que solicita la aplicación y pregunta al usuario si desea concederlos. Para cargar el audio se empleó un hilo (*thread*) secundario en lugar del hilo principal. De este modo se evita bloquear la GUI mientras dure el proceso. Implementar la carga de audio tuvo sus complicaciones debido a que las librerías comúnmente usadas para leer archivos de audio con Python no están compiladas para Android. Además, el proceso de compilación de cada una de ellas difiere en varios aspectos y complejidad. Se evaluaron las siguientes alternativas: pydub, soundfile, wave y librosa. Primero, el módulo wave es el

más sencillo, sin embargo solo permite subir audios wave en codificación PCM y no aquellos codificados como flotantes de 32 bits. Se evaluó la posibilidad de modificar este módulo para que soporte cargar audios con esta codificación, sin embargo, debido al tiempo requerido se optó por otra solución. El módulo pydub fue una excelente alternativa, sin embargo hubo dificultades con dos de las dependencias necesarias: FFmpeg y FFprobe, que son colecciones de librerías escritas en C. Por otro lado, la librería librosa también emplea muchas dependencias que no están compiladas para Android, por lo cual se descartó (por este motivo también se implementó la extracción de MFCCs con funciones propias). El módulo soundfile, basado en la librería libsndfile escrita en C, fue la alternativa ganadora dado que fue la primera que se consiguió compilar e integrar exitosamente para Android. Esto fue posible dado que en este caso se encontró información útil acerca de cómo compilarla empleando la herramienta CMake. Además, una vez compilada esta librería, se referenció a la misma en el archivo *buildozer.spec*, de modo que sea incluida durante la compilación y empaquetado de la aplicación. Cabe destacar que libsndfile debe ser compilada como una librería dinámica (extensión .so) dado que esto facilita mucho su integración en la aplicación. La otra alternativa era compilarla a modo de librería estática (extensión .a), pero ello requería enlazarla directamente en el binario final de la aplicación mediante un script personalizado debido a que Buildozer no cuenta con una opción directa para incluirla desde el archivo *buildozer.spec*. Un aspecto importante que queda por destacar de esta etapa, es que libsndfile es un conjunto de librerías de código abierto, entre ellas codificadores y decodificadores (como mp3lame y mpg123 respectivamente) para distintos formatos de audio (como MP3, M4A, entre otros). El módulo básico permite subir audios en formato WAV (sin compresión), pero para formatos comprimidos como los mencionados anteriormente se requiere de compilar libsndfile en conjunto con dependencias extra, lo cual escapa al alcance de este Proyecto y por lo tanto no se realizó. En definitiva, el soporte está limitado a audios en formato WAV.

9.4.2.3. Integración de Modelos

La tercera etapa se inició con la implementación de un modelo preliminar. Debido a que una vez desarrollada la lógica puede cambiarse fácilmente e integrar otros modelos semejantes, independizando esta actividad de las demás. Lo más importante fue contar con un ejemplo útil y bien documentado en un repositorio público en la plataforma GitHub [75]. Este código fue usado para el desarrollo de la aplicación. El mismo emplea PyJNIus para cargar los componentes de Tensor Flow Lite desde clases de Java a clases de Python. No hubo complicaciones importantes para esta etapa más allá del tiempo necesario para descargar las dependencias de Tensor Flow Lite en el entorno de desarrollo. Además, esta etapa incluyó la lógica para realizar las inferencias en base a un audio previamente cargado en memoria, para lo cual se empleó principalmente el módulo NumPy. Esta lógica también emplea un hilo secundario.

9.4.2.4. Mejora de la Pantalla de Muestra de Resultados

Esta etapa consistió principalmente en mostrar de forma intuitiva los porcentajes de probabilidad asignados por el modelo a cada clase según el audio proporcionado. Luego de evaluar alternativas, se optó por emplear un gráfico de barras horizontales para cada porcentaje de cada clase, y ordenar desde la clase más probable a la menos probable.

9.4.2.5. Grabación de Audio

La quinta etapa consistió en incorporar funcionalidad para grabar audio. Nuevamente se recurrió a PyJNIus para hacer uso de clases de Java con Python y así grabar y almacenar el audio. Esta etapa fue relativamente sencilla. Lo más difícil fue lidiar con los permisos a la hora de almacenar el audio. Al final, se optó por emplear el directorio "media", en el Almacenamiento Interno del dispositivo Android (concretamente en el directorio *"Almacenamiento Interno/Android/media/org.proyic.qbmapp"*). Esta ubicación en el sistema de archivos

está pensada para que las aplicaciónes almacenen archivos multimedia y no requiere permisos ni gestión especiales. Se incorporó en el archivo *buildozer.spec* los permisos `RECORD_AUDIO` y `WRITE_EXTERNAL_STORAGE`. Se destaca que esta funcionalidad no es esencial para la aplicación dado que puede emplearse cualquier otra aplicación para grabar el audio y luego subirlo desde la pantalla "subir audio".

9.4.2.6. Mejora de Aspecto Visual

La sexta etapa consistió en mejorar el aspecto visual añadiendo funcionalidad para cambio de tema, *presplash* (pantalla de carga de la aplicación), íconos y tamaño de fuente apropiado.

9.4.3. Funcionamiento de la aplicación

En la figura 9.13 se muestra el diagrama de flujo que corresponde a la etapa de inferencia en base a un audio previamente cargado en la memoria RAM, ya sea desde el almacenamiento interno mediante la opción "subir audio" o una nueva grabación mediante la opción "grabar audio". El audio, una vez cargado, pasa por un procesamiento de remuestreo y normalización de amplitud. Inmediatamente se recorta dicho audio en segmentos de 10 segundos de duración, y para cada uno de ellos se calculan y promedian los MFCCs. Luego, para cada vector de MFCCs se realiza la inferencia del modelo detector de abejas. Si este modelo detecta abejas en el audio, entonces pasa los vectores de MFCCs al modelo clasificador, de lo contrario, solo informa al usuario que no se han encontrado abejas. Se destaca que el procesamiento realizado en la etapa de inferencia de la aplicación es prácticamente el mismo que se realizó para la obtención de características durante el entrenamiento de los modelos.

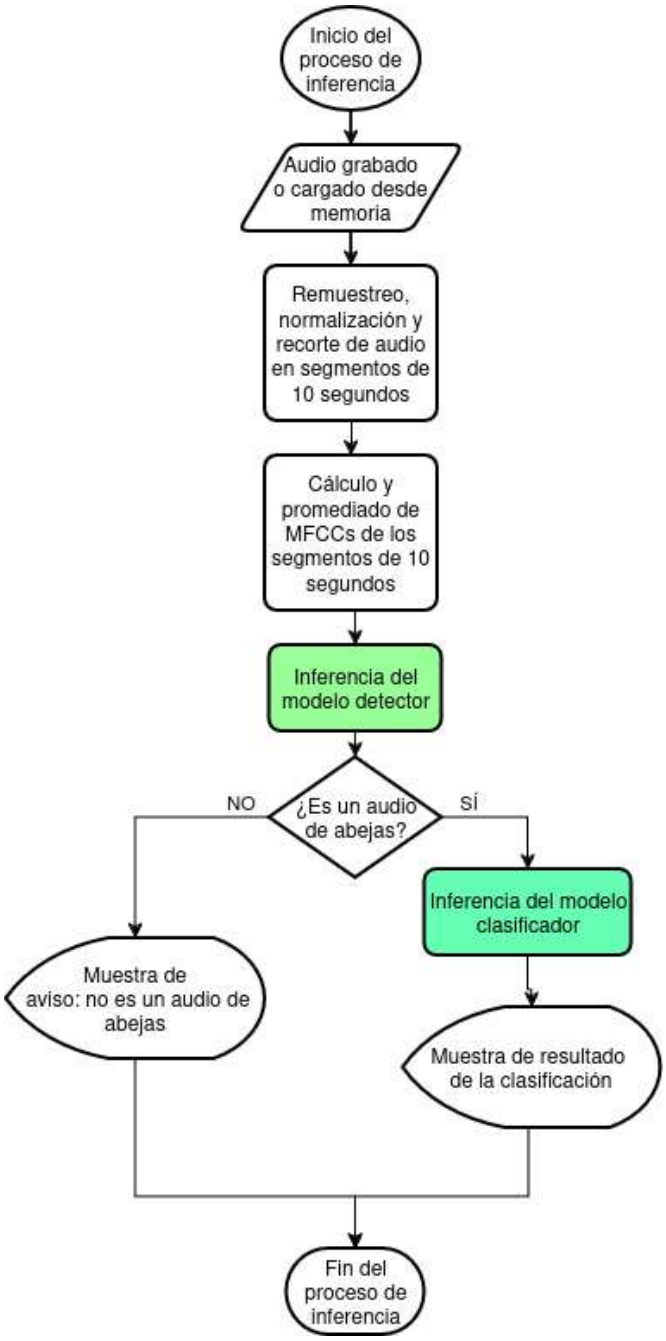

Figura 9.13: Diagrama de flujo simplificado del proceso de inferencia de la aplicación

9.5. Actividades en Campo

9.5.1. Obtención de audios

Se recopilaron grabaciones de sonidos de colmenas de la zona de 25 de Mayo, Misiones, Argentina, junto con audios de ruido ambiente para diversificar y enriquecer los datos utilizados en el entrenamiento del modelo detector. Se obtuvo audios del sonido de 10 colmenas. El 70 % de los audios fueron destinados al entrenamiento y lo restante al testeo preliminar. A continuación se presentan imágenes y descripciones de las consideraciones tomados en campo.

Se realizaron pruebas con 4 tipos de micrófonos, 3 integrados en los dispositivos móviles y un micrófono Lavalier corbatero con una frecuencia de muestreo de 16 kHz. En la figura 9.14(a) se puede ver el micrófono corbatero ubicado sobre la tapa de la colmena, también sobre la piquera [1] (figura 9.14(b) y un dispositivo móvil en la figura 9.14(c) sobre la piquera.

(a) Micrófono sobre tapa (b) Micrófono en piquera (c) Dispositivo en piquera

Figura 9.14: Ubicaciones de micrófonos

[1] Agujero o puerta pequeña que se hace en las colmenas para que las abejas puedan entrar y salir [76].

Ademas, se hicieron pruebas de grabaciones dentro de la colmena en la cámara de cría, tal como se observa en la figura 9.15. Cabe aclarar que luego de posicionar el micrófono donde indica dicha figura se cerró la tapa durante el tiempo de grabación.

Figura 9.15: Micrófono sobre cámara de cría

De las ubicaciones probadas y mostradas anteriormente, la grabación más representativa y con más información para el modelo sería la realizada en la cámara de cría, tal como se ve en la figura 9.15. La información de interés se encuentra en el nido de la colmena, lo que proporciona un comportamiento representativo del enjambre. Otro motivo por el cual se define este lugar de grabación es que los demás *dataset* utilizados en este trabajo también realizan las grabaciones en la cámara de cría. Realizar grabaciones desde el exterior de la colmena no es recomendable, ya que existen ruidos externos (pájaros, viento, vehículos), y el ruido de las abejas obreras entrando y saliendo de la piquera, que pueden llegar a confundir a los modelos y dificultar su trabajo.

Otro de los trabajos realizados en el campo fue la búsqueda de una reina. Para encontrarla, se realizó una búsqueda con poco humo durante aproximadamente 1 hora.

En la figura 9.16 se puede ver cómo fue necesario sacar todos los cuadros presentes en la colmena y revisar minuciosamente cada uno. Luego de encontrarse la reina, se empleó una jaula para reina, donde se puede observar cómo las abejas nodrizas y obreras rodean a la misma (figura 9.17(a)). Después de 5 minutos en la jaula, se procedió a liberarla en el piso de la colmena, tal como se ve en la figura 9.17(b).

Figura 9.16: Colmena sin cuadros

(a) Reina enjaulada (b) Reina liberada

Figura 9.17: Reina de la colmena

Una vez encontrada la reina, se procedió a eliminarla como se ve en la figura 9.18, dejándola para que las obreras detecten su ausencia y así puedan crear una nueva reina.

Figura 9.18: Reina eliminada

Esta colmena fue grabada al término de 7 días, luego de la eliminación de la reina, donde estos audios fueron recolectados para la etapa de comprobación de la aplicación con una colmena sin reina. Se detalla más de esto en la sección 9.5.2.

Además de la búsqueda de reina explicada anteriormente, se realizó una búsqueda en una colmena que presentaba un excesivo número de zánganos, lo cual es indicador de que la reina es muy vieja y poco fértil. La búsqueda de la reina de esta colmena se llevó a cabo durante las 3 visitas realizadas al apiario, donde se podía observar una mala postura de huevos y una mala organización de los alimentos dentro de la colmena. En ninguna de las visitas realizadas fue posible encontrar a la reina. En términos de horas de búsqueda de esta reina, aproximadamente entre 5 y 6 horas. En la última visita durante la búsqueda, otra colmena detectó la debilidad de la misma y realizó un pillaje (ataque para robar alimento), produciendo la muerte de la colmena completa.

9.5.2. Actividades de Validación

9.5.2.1. Validación en laboratorio

Las pruebas en ambiente controlado consistieron en tomar los audios apartados en test y verificar la capacidad del modelo para clasificarlos. En primer lugar se verificó el modelo detector, el cual fue testeado en todos los casos posibles, incluyendo la funcionalidad de subir un audio o grabar el sonido.

En la figura 9.19(a) se muestra la pantalla resultante tras realizar una inferencia en el caso en que el modelo detector no identifique que el audio corresponde al sonido de abejas. Por el contrario, en la figura 9.19(b) se presenta el resultado cuando el modelo sí detecta la presencia del sonido de abejas en el audio.

Los audios pueden ser grabados mediante la aplicación "Queen Bee Monitor" o cualquier otra aplicación que obtenga grabaciones y luego subirlo para su inferencia. Cabe destacar que deben ser guardados en formato WAV. Esto permitió comprobar el funcionamiento de la integración del modelo detector y la aplicación.

Para poder testear el funcionamiento e integración del modelo multiclase con la aplicación se utilizaron audios destinados al testeo del modelo, que como son audios de colmenas en los que se representa el estado de la abeja reina, pasan la inferencia del modelo detector y el modelo multiclase realiza la inferencia final, tal como se ve en la figura 9.19(b).

(a) No Bee (b) Bee

Figura 9.19: Integración del modelo detector

9.5.2.2. Validación en campo

La validación de campo consistió en emplear la aplicación en el mismo apiario en el cual previamente se recolectaron grabaciones de audio probando distintas ubicaciones de micrófono. En este caso se grabaron los audios directamente desde la aplicación y se realizó cada inferencia *in situ*. Esto permitió validar tanto el desempeño del modelo como la usabilidad de la aplicación en general. Se probaron casos de colmenas con reina y sin reina para validar el modelo. Se probaron 2 colmenas en las cuales se sabía que la reina estaba presente, y el modelo clasificó correctamente. Este resultado puede observarse

en la figura 9.20(a) y 9.20(b). Además, se probó con una grabación en la colmena para la cual se había eliminado a la reina. El audio fue subido desde el almacenamiento del dispositivo mediante el botón "subir audio". El modelo también clasificó adecuadamente este caso, marcando el estado de la reina como ausente.

(a) Grabación en proceso (b) Resultado

Figura 9.20: Validación en campo. Colmena con reina presente.

No se realizaron pruebas de validación en campo para el caso de inserción manual de reina debido a que no se contó con la posibilidad de llevar a cabo dicho procedimiento, por lo tanto las pruebas para el estado de la reina presente “aceptada” y “rechazada” solamente se hicieron pruebas en laboratorio.

Parte III

Conclusiones

CAPÍTULO 10

Conclusiones

10.1. Conclusión

Se concretó el objetivo general logrando desarrollar una aplicación móvil e integrar un modelo de ML capaz de determinar el estado de la abeja reina. A partir de un desarrollo integral que abarca desde la etapa de procesamiento de los datos hasta la obtención del APK.

La extracción de características mediante MFCCs permitió representar información útil de los audios de manera compacta y efectiva. La implementación del algoritmo para calcular estos coeficientes demostró ser exitosa y además tiene la ventaja de estar optimizada a nivel de requerimientos de

memoria debido a que solo fue necesaria la librería NumPy, que es fácilmente integrable a la app. Además de esto, el uso de SMOTE para el balanceo de clases permitió salvar las brechas entre la cantidad de ejemplos de cada estado de la abeja reina; contemplando además la corrección de etiquetas y la eliminación de *outliers*.

El uso de una red neuronal para la arquitectura de los modelos permitió optimizar el peso de los mismos haciendo una comparativa de las técnicas de optimización como pre-entrenamiento, pos-entrenamiento, *pruning* y agrupamiento de *clúster*, a la vez que se empleó la librería Tensor Flow Lite para integrarlos en la aplicación. Otra de las optimizaciones logradas fue mediante el uso de algoritmos genéticos para encontrar la topología adecuada. El modelo clasificador logró una precisión máxima del 98,06 % en el test, mientras que el modelo detector logró una precisión del 99,37 %.

El desarrollo de la aplicación fue un éxito, ya que se eligió una arquitectura modular, separando el código que corresponde a la lógica de aquel que define la GUI. Esto facilitó la implementación de mejoras graduales en el código y la escalabilidad de la aplicación. Por otro lado, la configuración del empaquetador Buildozer permitió ahorrar tiempo de compilación y reducir el tamaño del APK. Esto fue muy evidente, por ejemplo, al seleccionar la arquitectura objetivo específica (arm64-v8a).

El proceso de integración de los modelos de ML en la app resultó sencillo, dado que el proceso está descrito en la documentación de Tensor Flow Lite y además existen diversos ejemplos en GitHub, tal y como se mencionó en la sección 9.

Durante las pruebas de campo, se evaluaron diversas ubicaciones del micrófono para determinar la más efectiva. Los resultados mostraron que las grabaciones realizadas directamente sobre la cámara de cría ofrecían los mejores resultados, ya que los conjuntos de datos disponibles se basan en grabaciones tomadas en esa zona específica de la colmena. Es importante destacar que las pruebas *in situ* no incluyeron los casos de inserción manual de la reina, tanto en el caso de reina rechazada como de reina aceptada, debido a la

complejidad y el tiempo requerido para llevar a cabo este procedimiento de manera efectiva.

Además de esto, las pruebas de campo demostraron la facilidad de uso de la aplicación, debida a su diseño intuitivo con botones y texto grandes, lo cual es una ventaja a la hora de trabajar con guantes en un apiario.

En general, se ha obtenido un resultado más que satisfactorio, y se han alcanzado todos los objetivos específicos. Contemplando el estudio financiero-económico se tiene que es proyecto es rentable para un inversionista que quiera adquirir el desarrollo y distribuir la aplicación Queen Bee Monitor.

10.2. Trabajo Futuro

Al ser un desarrollo integral que abarca varias etapas, tecnologías y conocimientos, existen aspectos que se pueden extender, así como también mejoras que aplicar en base a los resultados obtenidos. Como trabajo futuro, se propone las siguientes subsecciones:

10.2.1. Mejora del modelo clasificador

Mejorar el modelo clasificador para el caso de inserción de nueva reina, específicamente entrenando con mejores datos, los cuales deberán obtenerse de alguna fuente distinta a la empleada en este proyecto.

10.2.2. Mejora de la aplicación móvil

- Extender la funcionalidad de la aplicación para que permita subir audios en otros formatos además de WAV.
- Mejorar la compatibilidad de la aplicación, realizando pruebas sobre un espectro más amplio de versiones de Android y dispositivos.

- Comprobar el espacio de almacenamiento del dispositivo antes de guardar nuevas grabaciones de audio.
- Añadir funcionalidad extra para el seguimiento de múltiples colmenas y apiarios.
- Implementar funcionalidad para realizar otros diagnósticos en base a sonido y/o fotografías.

Parte IV

Anexos

Análisis económico-financiero

Para este proyecto se desarrolló un análisis económico-financiero, teniendo en cuenta que se elabora un software a medida y para un usuario específico (apicultores). La inversión inicial corresponde a los costos presentados en la tabla 10.1, los cuales corresponden a bienes a adquirir. Los precios fueron extraídos de la plataforma de Mercado Libre [77] (Accedido el 16-09-2024).

A partir de la tabla 10.1 se procedió a realizar una comparativa con precios de aplicaciones que son destinadas a la actividad apícola. Si bien hay muchas aplicaciones para apicultura, se tuvo como referencia aquellas que implementan modelos de *machine learning*. De este modo se definió un precio de venta de $31,35$ *USD*. Se estableció un margen de beneficio del 80 % para cada una de las producciones de software. En cuanto al estudio de mercado realizado en 6, se plantea abarcar 2000 productores apícolas por año con la venta de la aplicación.

Tabla 10.1: Bienes a Adquirir

Bienes a Adquirir para el proyecto			
Costo total bienes a adquirir			**$3.824.600**
Maquinarias, equipos e instrumentos			
Descripción	**Cantidad**	**Precio unitario**	**Costo total por tipo**
Telefono Celular Moto G200	1	$520.000	$520.000
Telefono Celular Samsung Galaxy J7 Prime	1	$150.000	$150.000
Telefono Celular Samsung Galaxy A15	1	$300.000	$300.000
Cable USB tipo C	2	$5.000	$10.000
Computadora Portátil	2	$1.200.000	$2.400.000
Cable USB tipo v8	2	$5.000	$10.000
Mouse inalámbrico	2	$8.800	$17.600
Subtotal:			$3.407.600
Infraestructura			
Descripción	**Cantidad**	**Precio unitario**	**Costo total por tipo**
Cuenta de Play Store	1	$25.000,00	$25.000,00
Subtotal:			$25.000
Otros			
Descripción	**Cantidad**	**Precio unitario**	**Costo total por tipo**
Traje apicola	2	$178.000	$356.000
Ahumador	1	$26.000	$26.000
Pinza	1	$10.000	$10.000
Subtotal:			$392.000

10.3. Flujo de caja

Para este proyecto se plantea un horizonte de planeamiento de 5 años. A continuación se presentan los flujos de caja, tanto para el proyecto como para el inversionista, donde para este último se toma como referencia un préstamo del banco nación con una tasa del 37 % de interés. Estas tablas básicamente muestran la evolución de los ingresos y egresos en un periodo determinado considerando que para dicho periodo el valor del dólar no varía.

Tabla 10.2: Flujo de caja del proyecto

Flujo de caja del PROYECTO						
Años	0	1	2	3	4	5
Ing.		$59.774.364	$59.774.364	$59.774.364	$59.774.364	$59.774.364
C. Var.		$0	$0	$0	$0	$0
C. Fijos Des.		- $25.841.000	- $25.841.000	- $25.841.000	- $25.841.000	- $25.841.000
Imp. 16 %		- $9.563.898	- $9.563.898	- $9.563.898	- $9.563.898	- $9.563.898
Amort. Intang.		- $6.641.596	- $6.641.596	- $6.641.596	- $6.641.596	- $6.641.596
Util. Antes Imp.		$17.727.870	$17.727.870	$17.727.870	$17.727.870	$17.727.870
Imp. 35 %		- $6.204.754	- $6.204.754	- $6.204.754	- $6.204.754	- $6.204.754
Inv. Inicial	$7.157.980					
Inv. Cap. Trab.	$26.050.000					
Util. Neta		$11.523.115	$11.523.115	$11.523.115	$11.523.115	$11.523.115
Amort. Intang.		$6.641.596	$6.641.596	$6.641.596	$6.641.596	$6.641.596
Recupero						$26.050.000
F. Caja	- $33.207.980	$18.164.711	$18.164.711	$18.164.711	$18.164.711	$44.214.711
F. Acum.	- $33.207.980	- $15.043.269	$3.121.443	$21.286.154	$39.450.865	$83.665.577

En el histograma de la figura 10.1 se puede observar el recupero para el proyecto, donde a partir del año cuarto empieza a generar ganancias. Y en el caso del flujo de caja del inversionista, figura 10.2 a partir del quinto año genera rentabilidad.

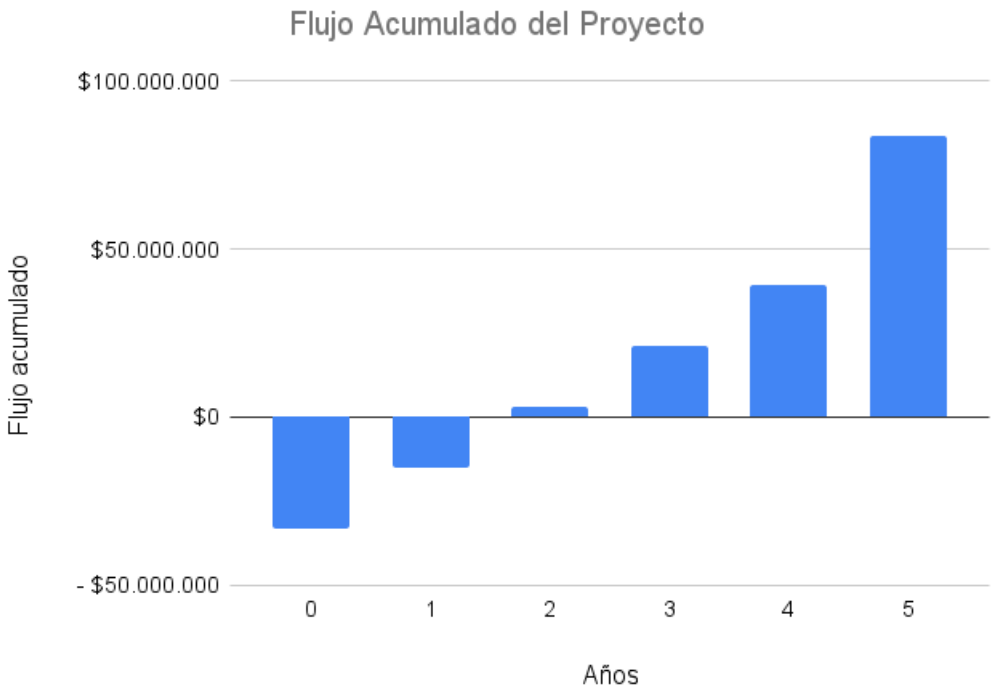

Figura 10.1: Flujo acumulado del proyecto.

Tabla 10.3: Flujo de caja del inversionista

Flujo de caja del INVERSIONISTA						
Años	0	1	2	3	4	5
Ing.		$59.774.364	$59.774.364	$59.774.364	$59.774.364	$59.774.364
Int. Prést.		- $8.678.681	- $7.929.274	- $6.857.621	- $5.325.157	- $3.133.735
C. Var.		$0	$0	$0	$0	$0
C. Fijos Des.		- $25.841.000	- $25.841.000	- $25.841.000	- $25.841.000	- $25.841.000
Imp. 16 %		- $9.563.898	- $9.563.898	- $9.563.898	- $9.563.898	- $9.563.898
Amort. Intang.		- $6.641.596	- $6.641.596	- $6.641.596	- $6.641.596	- $6.641.596
Util. Antes Imp.		$9.049.188	$9.798.596	$10.870.249	$12.402.712	$14.594.135
Imp. 35 %		- $12.216.404	- $13.228.104	- $14.674.836	- $16.743.662	- $19.702.082
Inv. Inicial	$7.157.980					
Inv. Cap. Trab.	$26.050.000					
Util. Neta		- $3.167.216	- $3.429.509	- $3.804.587	- $4.340.949	- $5.107.947
Amort. Intang.		$6.641.596	$6.641.596	$6.641.596	$6.641.596	$6.641.596
Recupero						$26.050.000
Prestamo	$20.182.980					
Amort. deuda		- $1.742.808	- $2.492.216	- $3.563.869	- $5.096.332	- $7.287.755
F. Caja	- $13.025.000	$7.613.544	$7.088.959	$6.338.802	$5.266.077	$29.782.082
F. Acum.	- $33.207.980	- $25.594.436	- $18.505.477	- $12.166.675	- $6.900.597	$22.881.484

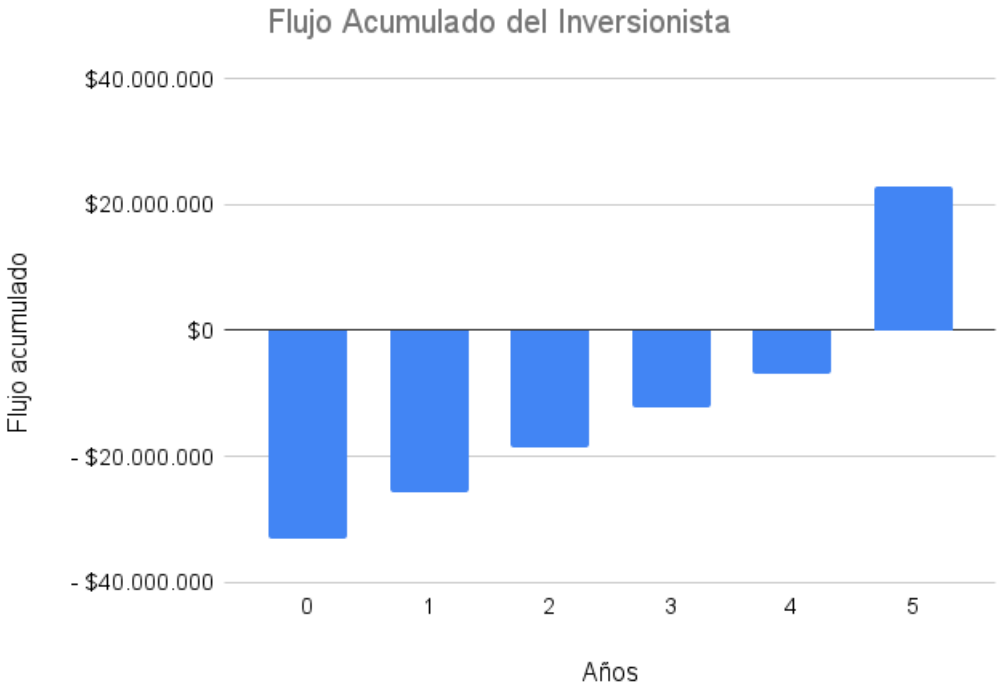

Figura 10.2: Flujo acumulado del inversionista.

10.3.1. Evaluación del proyecto

Para la evaluación del proyecto se aplica dos criterios, el de valor actual neto (VAN) y el de la tasa interna de retorno (TIR). Donde si TIR>TREMA y VPN>0, entonces el proyecto es rentable para el inversionista.

10.3.1.1. Método del valor actual neto (VAN)

Para determinar éste valor, se usa la tasa de retorno mínima atractiva (TREMA) para la que se tiene en cuenta la inflación del país y un índice de riesgo. Dado que el incremento anual por plazo fijo en el Banco Nación es de 37 %, se decide tomar un valor de TREMA, para el proyecto de 40 %. Mediante los cálculos correspondiente se tienen los valores del VAN:

$$VAN = \$6{,}145{,}555\ (Proyecto)$$

$$VAN = \$3{,}748{,}884\ (Inversionista)$$

10.3.1.2. Tasa interna de retorno (TIR)

Es la tasa de interés más alta que un inversionista puede pagar sin perder dinero.

$$TIR = 53\,\%\ (Proyecto)$$

$$TIR = 59\,\%\ (Inversionista)$$

10.3.1.3. Conclusiones parciales

En base a los resultados anteriores se tiene que el proyecto es rentable y cumple con la condición para una rentabilidad de un inversionista, ya que el TIR del inversionista es mayor a la TIR del proyecto. Esto se debe al apalancamiento que se produce cuando se pone en funcionamiento un proyecto que requiere una alta inversión inicial con un préstamo que es menor a la misma.

Por último, se destaca que gran parte de la inversión ya está disponible: recursos humanos, smartphones, PCs, cables, trajes apícolas y herramientas para las pruebas de campo e incluso apiario de colmenas. Los montos establecidos son una referencia concreta si se desea transferir el desarrollo.

Parte V

Referencias

Bibliografía

[1] «Apicultura,» visitado 15 de mayo de 2024. dirección: https://es.wikipedia.org/w/index.php?title=Apicultura&oldid=159617871.

[2] E. Guzmán-Novoa, A. C. Benítez y L. G. E. Montaño, «Colonización, impacto y control de las abejas melíferas africanizadas en México,» *Veterinaria México*, vol. 42, n.º 2, págs. 149-178, jun. de 2011.

[3] R. A. O. M. Sarmiento, *Manual técnico de apicultura : abeja (Apis mellifera)*, 2012.

[4] D. Howard, O. Duran, G. Hunter y K. Stebel, «Signal processing the acoustics of honeybees (APIS MELLIFERA) to identify the "queenless" state in Hives,» *Proceedings of the Institute of Acoustics*, vol. 35, págs. 290-297, ene. de 2013.

[5] C. Uthoff, M. N. Homsi y M. von Bergen, «Acoustic and vibration monitoring of honeybee colonies for beekeeping-relevant aspects of presence of queen bee and swarming,» *Computers and Electronics in Agriculture*, vol. 205, pág. 107 589, feb. de 2023.

[6] M. Abdollahi, P. Giovenazzo y T. H. Falk, «Automated Beehive Acoustics Monitoring: A Comprehensive Review of the Literature and Recommendations for Future Work,» *Biology*, 2022. visitado 4 de jul. de 2024. dirección: https://www.mdpi.com/2076-3417/12/8/3920.

[7] «La apicultura posiciona al país como uno de los principales exportadores mundiales de miel,» visitado 15 de jun. de 2024. dirección: https://agro.misiones.gob.ar/2023/06/27/la-apicultura-posiciona-al-pais-como-uno-de-los-principales-exportadores-mundiales-de-miel/.

[8] «Inteligencia artificial en la apicultura: así está cambiando el trabajo apícola,» visitado 30 de jun. de 2024. dirección: https://apiculturaymiel.com/abejas/inteligencia-artificial-en-la-apicultura-dispositivos-aplicaciones-cambiaran-mundo-apicola/.

[9] «Beewise is saving bees to protect the global food supply,» visitado 29 de jun. de 2024. dirección: https://beewise.ag/home.

[10] M. E. Verlag, «Varroosis de las Abejas Melíferas,» *World Organisation for Animal Health*, 2021. visitado 25 de sep. de 2024. dirección: https://www.woah.org/fileadmin/Home/esp/Health_standards/tahm/3.02.07_Varroosis.pdf.

[11] D. Kanelis, «Decoding the Behavior of a Queenless Colony Using Sound Signals,» *Biology*, vol. 12, n.º 11, pág. 11, nov. de 2023.

[12] D. I. Kiromitis y C. V. Bellos, «Bee Sound Detector: An Easy-to-Install, Low-Power, Low-Cost Beehive Conditions Monitoring System,» *Electronics*, vol. 11, 2022. visitado 25 de sep. de 2024. dirección: https://www.mdpi.com/2079-9292/11/19/3152.

[13] «OSBeehives,» visitado 30 de jun. de 2024. dirección: https://valldaura.net/osbeehives/.

[14] I. Nolasco, A. Terenzi y S. Cecchi, «Audio-Based identification of Beehive states: The dataset,» *Zenodo*, feb. de 2019. visitado 28 de jun. de 2024.

[15] «Transfer learning with YAMNet for environmental sound classification,» visitado 4 de jul. de 2024. dirección: https://www.tensorflow.org/tutorials/audio/transfer_learning_audio.

[16] «Abejas y colmenas sanas,» visitado 30 de jun. de 2024. dirección: https://www.beescanning.com/es/.

[17] «ziBees – Product identification. »dirección: https://www.behance.net/gallery/68997413/ziBees-Product-identification.

[18] «Varroa Counter - Aplicaciones en Google Play,» visitado 29 de jun. de 2024. dirección: https://play.google.com/store/apps/details?id=topLab.VarroaCounter&hl=es.

[19] «Bee Queen Detector - Aplicaciones en Google Play,» visitado 30 de jun. de 2024. dirección: https://play.google.com/store/apps/details?id=org.bqd.queen_detector&hl=es.

[20] «Bee Health Guru,» visitado 28 de jun. de 2024. dirección: https://www.kickstarter.com/projects/beehealthguru/bee-health-guru-a-smartphone-app-for-beekeepers.

[21] «¿Qué es la apicultura?» Visitado 22 de jun. de 2024. dirección: http://www.gob.mx/agricultura/es/articulos/que-es-la-apicultura.

[22] «El Zumbido de las Abejas y la Comunicación de los Enjambres,» visitado 4 de jul. de 2024. dirección: https://www.apicolturalaterza.com/es/el-zumbido-de-las-abejas-y-la-comunicacion-de-los-enjambres.

[23] «Trasiego de Colmenas – Alimentos Para Los Agricultores,» visitado 29 de jun. de 2024. dirección: http://food4farmers.org/es/2015/02/23/trasiego-de-colmenas/.

[24] «20 de mayo Día Mundial de la Abeja: LA APICULTURA ESTÁ PRESENTE EN LOS 76 MUNICIPIOS DE MISIONES,» visitado 15 de jun. de 2024. dirección: https://agro.misiones.gob.ar/2020/05/20/el-20-de-mayo-dia-mundial-de-la-abeja-la-apicultura-esta-presente-en-los-76-municipios-de-misiones/.

[25] *Apicultura*, 2021. visitado 22 de oct. de 2024. dirección: https://magyp.gob.ar/apicultura/.

[26] A. M. IESC, «Artificial Intelligence, Machine Learning, and Deep Learning: Same context, Different concepts,» *IESC*, 2018. visitado 4 de jul. de 2024.

[27] «¿Qué es la Inteligencia Artificial?» Visitado 4 de jul. de 2024. dirección: http://nic.ar/es/enterate/novedades/que-es-inteligencia-artificial.

[28] P. R. Ramírez, «Clasificación automática de sonidos utilizando aprendizaje máquina,» Tesis doct., Dpto. Teoría de la Señal y Comunicaciones Escuela Técnica Superior de Ingeniería Universidad de Sevilla, 2020. visitado 19 de oct. de 2024. dirección: https://biblus.us.es/bibing/proyectos/abreproy/92880/fichero/TFG-2880+RODR%C3%8DGUEZ+RAM%C3%8DREZ%2C+PATRICIO.pdf.

[29] *Dominio de la frecuencia - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Dominio_de_la_frecuencia, 2005. visitado 18 de oct. de 2024.

[30] *Frecuencia de muestreo - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Frecuencia_de_muestreo, 2014. visitado 18 de oct. de 2024.

[31] *Teorema de muestreo de Nyquist-Shannon - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon, 2016. visitado 18 de oct. de 2024.

[32] W. Storr, *Low Pass Filter - Passive RC Filter Tutorial — electronics-tutorials.ws*. visitado 18 de oct. de 2024. dirección: https://www.electronics-tutorials.ws/filter/filter_2.html.

[33] G. Brown, *Digital Audio Basics: Audio Sample Rate and Bit Depth*, 2021. visitado 18 de oct. de 2024. dirección: https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html.

[34] *Normalización de audio - Wikipedia, la enciclopedia libre — es.wikipedia.org*, 2024. visitado 18 de oct. de 2024. dirección: https://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_audio#:~:text=La%20normalizaci%C3%B3n%20de%20audio%20es,se%C3%B1al%2Druido%20generalmente%20no%20cambia..

[35] *Espectrograma - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Espectrograma, 2015. visitado 18 de oct. de 2024.

[36] *Escala Mel - Wikipedia, la enciclopedia libre — es.wikipedia.org*, 2024. visitado 19 de oct. de 2024. dirección: https://es.wikipedia.org/wiki/Escala_Mel.

[37] *Mel-frequency cepstrum - Wikipedia — en.wikipedia.org*, 2024. dirección: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum.

[38] *Mel Filter Bank*, 2014. visitado 22 de oct. de 2024. dirección: https://siggigue.github.io/pyfilterbank/melbank.html.

[39] *EXTRACCIÓN DE CARACTERÍSTICAS*, 2020. visitado 19 de oct. de 2024. dirección: https://biblus.us.es/bibing/proyectos/abreproy/12054/fichero/MEMORIA%252F8.Cap%C3%ADtulo+3.pdf.

[40] F. M. Aguirre, «Desarrollo y análisis de clasificadores de señales de audio,» Tesis doct., 2017. visitado 21 de oct. de 2024. dirección: https://riunet.upv.es/bitstream/handle/10251/90005/Aguirre%20-%20Desarrollo%20y%20an%C3%A1lisis%20de%20clasificadores%20de%20se%C3%B1ales%20de%20audio..pdf?sequence=1.

[41] *Android | Do More With Google on Android Phones & Devices — android.com*, https://www.android.com/. visitado 1 de oct. de 2024.

[42] *Android - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Android. visitado 1 de oct. de 2024.

[43] *API - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/API. visitado 1 de oct. de 2024.

[44] *Kit de desarrollo de software - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software. visitado 1 de oct. de 2024.

[45] *Cómo comenzar a usar el NDK | Android NDK | Android Developers — developer.android.com*, https://developer.android.com/ndk/guides?hl=es-419. visitado 1 de oct. de 2024.

[46] *Framework - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Framework. visitado 1 de oct. de 2024.

[47] «Kivy: Cross-platform Python Framework for NUI,» visitado 24 de jun. de 2024. dirección: https://www.kivy.org/.

[48] «KivyMD 2.0.1.dev0 documentation,» visitado 28 de jun. de 2024. dirección: https://kivymd.readthedocs.io/en/latest/.

[49] «Material Design,» visitado 28 de jun. de 2024. dirección: https://m3.material.io/.

[50] *Welcome to Buildozer 2019 documentation*, 2019. visitado 28 de oct. de 2024. dirección: https://buildozer.readthedocs.io/en/latest/.

[51] *Entorno de desarrollo integrado - Wikipedia, la enciclopedia libre — es.wikipedia.org*, https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado. visitado 1 de oct. de 2024.

[52] *Acerca de GitHub y Git - Documentación de GitHub — docs.github.com*, https://docs.github.com/es/get-started/start-your-journey/about-github-and-git, 2024. visitado 1 de oct. de 2024.

[53] *GitHub: Let's build from here — github.com*, https://github.com/, 2024. visitado 1 de oct. de 2024.

[54] «Apis mellifera | Ecos del Bosque,» visitado 15 de jun. de 2024. dirección: https://ecosdelbosque.com/fauna/apis-mellifera.

[55] A. Yang, *Smart Bee Colony Monitor: Clips of Beehive Sounds*. visitado 28 de jun. de 2024. dirección: https://www.kaggle.com/datasets/annajyang/beehive-sounds.

[56] *SMOTE: synthetic minority over-sampling technique: Journal of Artificial Intelligence Research: Vol 16, No 1 — dl.acm.org*, https://dl.acm.org/doi/10.5555/1622407.1622416. visitado 27 de nov. de 2024.

[57] *Bee Dataset BUT-2*. visitado 20 de sep. de 2024. dirección: https://www.kaggle.com/datasets/imonbilk/bee-dataset-but-2/data.

[58] *Bee Audio Object Detection*. visitado 19 de sep. de 2024. dirección: https://www.kaggle.com/datasets/andrewlca/bee-audio-object-detection/data.

[59] *ESC-50 dataset*. visitado 6 de sep. de 2024. dirección: https://github.com/trinhtuanvubk/tflite-yamnet-audio-classification/blob/main/README.md.

[60] *AudioSet*. visitado 16 de oct. de 2024. dirección: https://research.google.com/audioset/dataset/index.html.

[61] *urbansound8k dataset*. visitado 24 de oct. de 2024. dirección: https://urbansounddataset.weebly.com/urbansound8k.html.

[62] *YouTube-8M Segments Dataset*. visitado 9 de oct. de 2024. dirección: https://research.google.com/youtube8m/.

[63] *YouTube-8M Tensorflow Starter Code*. visitado 9 de oct. de 2024. dirección: https://github.com/google/youtube-8m/tree/master.

[64] *A feature-rich command-line audio/video downloader*, 2024. visitado 27 de nov. de 2024. dirección: https://pypi.org/project/yt-dlp/.

[65] M. Ali, *PyCaret: An open source, low-code machine learning library in Python.* 2020. visitado 20 de nov. de 2024. dirección: https://www.pycaret.org.

[66] C. G. Northcutt, L. Jiang e I. L. Chuang, «Confident Learning: Estimating Uncertainty in Dataset Labels,» *Journal of Artificial Intelligence Research (JAIR)*, vol. 70, págs. 1373-1411, 2021.

[67] PyPI y Conda, *CleanLab: cleanlab open-source documentation*, 2024. visitado 21 de nov. de 2024. dirección: https://docs.cleanlab.ai/.

[68] A. A. Skrauba, H. D. Sosa y S. D. Zakowicz, *Machine Learning en la Apicultura: Clasificación Avanzada del Estado de Colmenas con Pipeline Optimizado y Etiquetado Preciso.* Presentado en las XIV Jornadas de Investigación, Desarrollo Tecnológico, Extensión, Vinculación y Muestra de la Producción (JIDeTEV) de la Facultad de Ingeniería de Oberá, Misones, Argentina, Disponible en: https://github.com/AxelSkrauba/queen-bee-status, ago. de 2024.

[69] S. Igemhokhai, *IMPLEMENTATION OF GENETIC ALGORITHMS(GA) FOR ENHANCED MODEL PERFORMANCE AND OPTIMAL RESULTS*, 2023. visitado 20 de oct. de 2024. dirección: https://medium.com/@i.v.shedrach/implementation-of-genetic-algorithms-ga-for-enhanced-model-performance-and-optimal-results-93c5e7a510d8.

[70] C. B. Egeli, *Optimizing Machine Learning Models with Genetic Algorithm-Based Hyperparameter Tuning*, 2024. visitado 21 de oct. de 2024. dirección: https://medium.com/@burak96egeli/optimizing-machine-learning-models-with-genetic-algorithm-based-hyperparameter-tuning-76d6f15fde6c.

[71] *ARM's latest CPUs push Android phone makers toward 64-bit only devices — engadget.com*, https://www.engadget.com/arms-latest-cpus-push-android-phone-makers-toward-64-bit-only-devices-165606654.html. visitado 18 de sep. de 2024.

[72] D. Whaley, *What 64-Bit Android Apps Mean for the Future of Mobile — newsroom.arm.com*, https://newsroom.arm.com/blog/android-64bit-future-mobile. visitado 18 de sep. de 2024.

[73] u/faze$_{f}azebook$, *¿Cuáles son las arquitecturas de CPU de Android más utilizadas?* https://www.reddit.com/r/AndroidQuestions/comments/y4z9hd/what_are_the_most_used_android_cpu_architectures/. visitado 18 de sep. de 2024.

[74] *Smartphones run 32-bit or 64-bit OS? — quora.com*, https://www.quora.com/Smartphones-run-32-bit-or-64-bit-OS. visitado 18 de sep. de 2024.

[75] R. D. Smith, *GitHub - teticio/kivy-tensorflow-helloworld: Run inference with Tensorflow Lite on iOS, Android, MacOS, Windows and Linux using Python. — github.com*, https://github.com/teticio/kivy-tensorflow-helloworld.

[76] Real Academia Española (RAE) y Asociación de Academias de la Lengua Española (ASALE), *Piquera | Diccionario de la lengua española*, 2024. visitado 22 de jun. de 2024. dirección: https://dle.rae.es/piquera.

[77] *Mercado Libre Argentina*, https://www.mercadolibre.com.ar/.

Formulario de autorización de depósito de tesis/trabajo final integrador en la Comunidad Ingenierías y Tecnologías del RIDUNaM (Repositorio Institucional Digital de la UNaM)

Por intermedio de la presente, el abajo firmante, AUTOR del TFI (Grado) titulada: **Machine Learning en Aplicación Móvil para la Clasificación del Estado de la Abeja Reina mediante Sonido**

Da FE de la autoría y originalidad de la obra mencionada, que fue dirigida por el Ing. Axel Skrauba

presentada y defendida en la Facultad de Ingeniería de la Universidad Nacional de Misiones (FI-UNaM), el (fecha) …17…/…12…. /…2024……, Acta/Expdte. N°183552 con el fin de obtener el título de Ingeniero en computación.

Tildar según corresponda
[ ] Tesis de Posgrado
[ ] Doctorado [ ] Maestría [X] **Trabajo Final Integrador**
[X] Tesis de Grado

Derechos patrimoniales
Como autor, expreso mi conformidad en cuanto a la cesión gratuita de los derechos de reproducción y circulación de esta obra, en forma NO EXCLUSIVA, a la Facultad de Ingeniería de Obera-UNaM. Dicha reproducción y circulación se podrá realizar, una o varias veces, en cualquier soporte, para todo el mundo, con fines sociales, educativos y científicos.
En virtud del carácter no exclusivo de esta cesión, el autor podrá reproducir y comunicar libremente la tesis o trabajo final integrador, a través de los medios que estime oportunos.

Condiciones de acceso en línea
[X] Autorizo el depósito del trabajo final integrador en forma inmediata
[ ] Autorizo el depósito del documento con embargo por el plazo de _____ meses a partir de la defensa de la misma.

Condiciones de uso TFI
Será puesta a disposición pública bajo las siguientes condiciones de uso:

	**(BY) Atribución** — Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciante (pero no de una manera que sugiera que tiene su apoyo o que apoyan el uso que hace de su obra).
	**(NC) No Comercial** — No puede utilizar esta obra para fines comerciales.
	**(SA) Permite trabajos derivados** — Siempre que se mantenga la misma licencia.
	**Reconocimiento – NoComercial – CompartirIgual (by-nc-sa):** No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Referencias:
CC (Licencias Creative Commons).
BY (Atribución).
NC (No comercial).
SA (Compartir igual).

Dados personales (llenar un cuadro por cada autor)

Apellido y Nombres	De Sosa, Héctor
Teléfono/Celular	3755-635727
Correo electrónico	hectordesosa.01@gmail.com

Apellido y Nombres	Zakowicz, Sandro Daniel
Teléfono/Celular	3755-272198
Correo electrónico	sandrodanielzakowicz@gmail.com

Se firma la presente en la Ciudad de Oberá, Misiones a los.....05..... días del mes de.....mayo..... de.....2026.....-

Zakowicz S.D. Zakowicz Sandro Daniel 43.357.486

De Sosa Héctor 43.529.213