elektron

Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026

Original Article

Clasificación acústica de cangrejos con modelos neuronales compactos

Acoustic classification of crabs with compact neural models

M. Celeste Cebedio [†‡], Martín Lorusso[*#], Leonardo Arnone[†‡], Lucas A. Rabioglio[†‡], Maximiliano Antonelli[†‡§], Raúl E. Lopresti[†‡§], Luciana De Micco[†‡§] y M. Paz Sal Moyano[*#§]

† Instituto de Investigaciones Científicas y Tecnológicas en Electrónica (ICYTE)
‡ Facultad de Ingeniería, Universidad Nacional de Mar del Plata (FI - UNMdP)
§ Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
∗ Instituto de Investigaciones Marinas y Costeras (IIMyC)
 # FCEyN, Universidad Nacional de Mar del Plata-CONICET

{celestecebedio,leoarn,lucas.rabioglio,maxanto,raul.lopresti,ldemicco,salmoyan}@fi.mdp.edu.ar

Received: 2025-12-16; Accepted: 2026-03-30

***Abstract*—This work presents an acoustic classifier based on neural networks, designed for implementation in a portable system for the field detection and classification of signals emitted by *Neohelice granulata* and *Cyrtograpsus angulatus*. From recordings acquired with a hydrophone in a controlled environment, prefiltering and adaptive segmentation techniques are applied to extract 22 acoustic features, which are used to train a low-dimensional neural network. Compression techniques, such as quantization, pruning, and knowledge distillation, are employed to obtain a compact model. Regarding knowledge distillation, different configurations of teacher models, segmentation windows, and sampling schemes are analyzed to produce an efficient model suitable for low-cost microcontrollers. The final model is exported in TensorFlow Lite format, ready for integration into embedded platforms, achieving an average class accuracy of 83.75 %, while maintaining low CPU and memory requirements compatible with low-cost embedded systems.**

***Keywords*— Bioacoustics, acoustic classification, neural networks, embedded systems, microcontrollers, knowledge distillation.**

***Resumen*— En este trabajo se presenta un clasificador acústico basado en redes neuronales, diseñado para su implementación en un sistema portátil para la detección y clasificación en campo de señales emitidas por *Neohelice granulata* y *Cyrtograpsus angulatus*. A partir de grabaciones adquiridas con un hidrófono en un entorno controlado, se aplican técnicas de prefiltrado y segmentación adaptativa para extraer 22 características acústicas, que se utilizan en el entrenamiento de una red neuronal de baja dimensionalidad. Se emplean técnicas de compresión, incluyendo cuantización, pruning y destilación del conocimiento, para obtener un modelo compacto y eficiente. En particular, se analizan distintas configuraciones de modelos maestros, ventanas de segmentación y esquemas de muestreo, con el objetivo de generar un modelo apto para microcontroladores de bajo costo. El modelo final se exporta en formato TensorFlow Lite, listo para su integración en plataformas embebidas, alcanzando una exactitud promedio entre clases del 83,75 %, mientras se mantienen bajos requerimientos de CPU y memoria.**

***Palabras clave*— Bioacústica, clasificación acústica, redes neuronales, sistemas embebidos, microcontroladores, destilación del conocimiento.**

I. Introducción

La bioacústica marina ha emergido como una herramienta poderosa para el estudio y la conservación de la biodiversidad marina [1]. La Reserva de Mar Chiquita, declarada reserva Mundial de la Biosfera por la UNESCO, es un ambiente estuarino de alta riqueza y diversidad biológica [2], [3]. Entre las especies clave de la Albufera se encuentran los cangrejos *Neohelice granulata* (Neo) y *Cyrtograpsus angulatus* (Cry). Estos crustáceos desempeñan roles fundamentales como ingenieros del ecosistema: sus actividades de excavación modifican el hábitat, influyen en la vegetación y afectan la disponibilidad de recursos para otras especies. En este contexto, es de vital importancia el estudio de los sonidos asociados a su comportamiento [4]. Sin embargo, el análisis manual de grabaciones extensas resulta inviable en estudios experimentales a gran escala, lo que impulsa la necesidad de desarrollar métodos automáticos de detección y clasificación.

Los recientes avances en inteligencia artificial (IA) y procesamiento digital de señales abren nuevas posibilidades para la aplicación de estas herramientas en el estudio de la biodiversidad [5]. El uso de tecnologías basadas en IA representa un campo en rápida expansión dentro de la investigación biológica [6], [7]. Por ejemplo, en [8] se utilizan redes convolucionales para clasificar especies de cangrejos con resultados prometedores, mientras que en [9] se aplican redes profundas sobre imágenes para objetivos similares. No obstante, un desafío importante de estas técnicas es su

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

elevado requerimiento de recursos computacionales, lo que dificulta su implementación en entornos de campo, como lagunas o zonas costeras, donde el uso de equipos de alto rendimiento resulta impráctico y costoso.

Las plataformas embebidas de bajo costo, como el ESP32 [10], [11] y los microcontroladores de la familia STM32 [12], se han consolidado como alternativas viables para la ejecución de modelos de IA en sistemas portátiles y de bajo consumo. Mediante el uso de modelos compactos y optimizados, estas plataformas permiten la implementación de redes neuronales en escenarios de campo, tal como ha sido reportado en trabajos recientes de TinyML [13].

En este trabajo se aborda el diseño de una red neuronal cuantizada y optimizada para su ejecución en plataformas embebidas de bajo costo. Se parte de grabaciones crudas adquiridas mediante un hidrófono, que posteriormente se procesan para construir un conjunto de datos etiquetado con tres clases de señales: Neohelice granulata, Cyrtograpsus angulatus y ruido ambiental. En la etapa de preprocesamiento se evalúan distintos esquemas de filtrado pasa-banda, así como técnicas de reducción de la tasa de muestreo y extracción de estadísticas simples, con el fin de analizar su impacto en el desempeño del clasificador. Asimismo, se analizan distintas arquitecturas de red y se emplea destilación del conocimiento junto con entrenamiento consciente de la cuantización, con el objetivo de reducir la complejidad del modelo manteniendo un rendimiento adecuado. Como resultado, se obtiene un modelo final en formato ligero, listo para su despliegue directo en una plataforma embebida de bajo costo.

II. Descripción general del sistema

El proyecto en el que se enmarca este trabajo tiene como objetivo el desarrollo de un sistema automatizado y de bajo costo capaz de procesar señales acústicas provenientes de un hidrófono en tiempo real, detectar la presencia de vocalizaciones de cangrejos y almacenar únicamente los segmentos relevantes. Este sistema está orientado a la identificación de sonidos producidos por *Neohelice granulata* y *Cyrtograpsus angulatus* en entornos con ruido ambiental.

A nivel general, el sistema completo, esquematizado en la Fig. 1, contempla múltiples etapas: adquisición de la señal acústica, filtrado pasa-banda, segmentación temporal, extracción de características, clasificación y gestión de memoria. Dentro de esta arquitectura, el módulo de clasificación neuronal cumple un rol central, ya que es el encargado de determinar la presencia o ausencia de eventos de interés y, en consecuencia, habilitar el almacenamiento de los segmentos acústicos correspondientes.

En este trabajo se aborda específicamente el desarrollo del clasificador neuronal y su flujo de construcción, independientemente de los mecanismos de adquisición y gestión de memoria, los cuales forman parte de etapas posteriores del sistema. En particular, se considera una cadena de procesamiento compuesta por una etapa de filtrado, seguida de la segmentación, el preprocesamiento y la extracción de características, cuyos resultados son utilizados como entrada del modelo de clasificación.

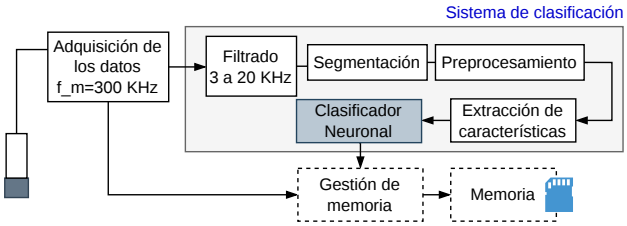


Figura 1: Esquema del sistema completo.

Cabe destacar que la arquitectura propuesta desacopla la etapa de adquisición de alta velocidad de la etapa de clasificación embebida. Esto se debe, por un lado, a que los registros acústicos de interés presentan duraciones del orden de varios minutos, mientras que el clasificador opera sobre ventanas de corta duración utilizadas como mecanismo de detección. Por otro lado, la tasa de muestreo requerida para el registro completo (del orden de cientos de kHz) excede las capacidades de adquisición sostenida de microcontroladores de bajo costo, como el ESP32. En consecuencia, se adopta un esquema en el cual el clasificador actúa como un disparador que habilita el almacenamiento de la señal en alta resolución en un sistema externo.

III. Arquitectura funcional del sistema de clasificación

El sistema de clasificación completo a implementar opera de forma continua a partir de las señales adquiridas por el hidrófono y se organiza en etapas claramente definidas.

En primer lugar, la señal acústica es adquirida y sometida a una etapa de filtrado pasa-banda, cuyo objetivo es restringir el contenido espectral a la banda de interés asociada a las vocalizaciones de los cangrejos. Esta etapa puede implementarse mediante un filtro digital en el microcontrolador o mediante un filtrado analógico previo.

A continuación, la señal filtrada se divide en segmentos temporales sobre los cuales se aplican operaciones de preprocesamiento simples, tales como cálculo de energía (Root Mean Square, RMS) o submuestreo, con el fin de reducir la cantidad de datos y resaltar información relevante.

Sobre estos segmentos se extrae un conjunto reducido de características temporales, mediante el extractor `catch22`, que constituyen la entrada del clasificador neuronal.

A partir de estas características, la red neuronal determina la presencia de un evento acústico y, en caso afirmativo, identifica la especie correspondiente.

El resultado de la clasificación se utiliza para controlar el sistema de almacenamiento, de modo que los datos crudos o los segmentos temporales asociados se guardan únicamente cuando se detecta un evento de interés. En la Fig. 2 se presenta un esquema funcional del sistema.

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

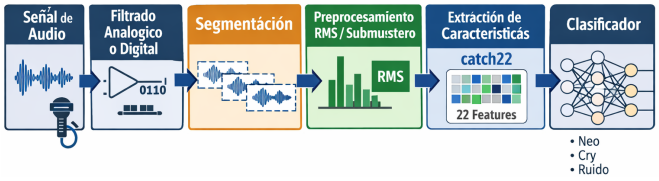


Figura 2: Esquema funcional del sistema completo de clasificación.

IV. Filtrado

Especialistas en el área indican que algunas especies de cangrejos podrían emitir señales con componentes de hasta 150 kHz. Sin embargo, tanto los estudios experimentales disponibles como el análisis empírico realizado en este trabajo muestran que la información relevante para la detección y discriminación se concentra por debajo de 20 kHz. En consecuencia, se adopta este valor como límite superior para el procesamiento del sistema de clasificación.

No obstante, dado que no puede descartarse completamente la presencia de componentes de mayor frecuencia, el sistema completo prevé el almacenamiento de las señales en su ancho de banda original, permitiendo su análisis posterior sin pérdida de información.

En función de este criterio, se define una etapa de filtrado pasa-banda que conserva únicamente la banda comprendida entre 3 kHz y 20 kHz, eliminando componentes de baja frecuencia asociadas al ruido ambiental y de altas frecuencias irrelevantes para la tarea de clasificación.

Para su implementación se evalúan dos enfoques: un filtrado digital en el microcontrolador y un filtrado analógico en una etapa externa.

- **Filtro digital.** Se implementa un filtro Butterworth pasa-banda de sexto orden como filtro IIR, diseñado para operar dentro de las restricciones de memoria y capacidad de cómputo del microcontrolador. Su respuesta en frecuencia presenta una atenuación adecuada fuera de banda y una implementación eficiente en términos de recursos, lo que lo hace compatible con la ejecución en tiempo real en hardware embebido.
- **Filtro analógico.** Para reducir la carga de procesamiento del microcontrolador, se evalúa un filtrado analógico mediante la simulación digital de un filtro Butterworth pasa-banda de décimo orden, implementado como un IIR en secciones de segundo orden (SOS) y aplicado con filtrado en fase cero. Este modelado se realiza con el objetivo de emular el comportamiento esperado de una implementación analógica equivalente, permitiendo su comparación directa con el filtrado digital en términos de desempeño del clasificador.

Para analizar el impacto sobre el desempeño global del sistema, se realizan simulaciones comparativas entre ambos filtros, utilizando como señal de entrada grabaciones crudas del sensor, correspondientes a registros acústicos de cangrejos obtenidos en condiciones controladas (véase Sección V-A).

Para el caso del filtrado analógico, los datos originales fueron convertidos a formato mono y normalizados antes del filtrado, de manera de reproducir con fidelidad la acción esperada de un filtro analógico equivalente aplicado en la etapa de entrada. En la Fig. 3 se muestra la transferencia del filtro utilizado y en la Fig. 4 la respuesta obtenida.

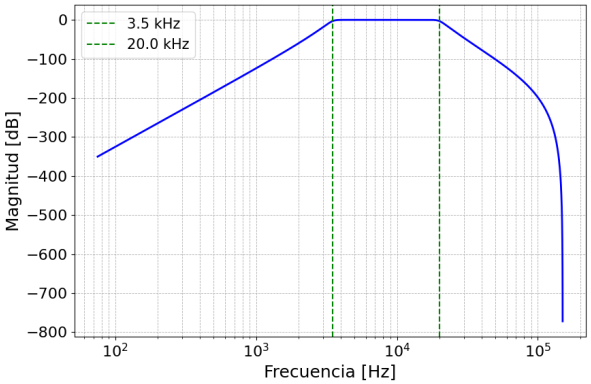


Figura 3: Respuesta en frecuencia del modelo digital equivalente al filtro analógico pasa-banda Butterworth.

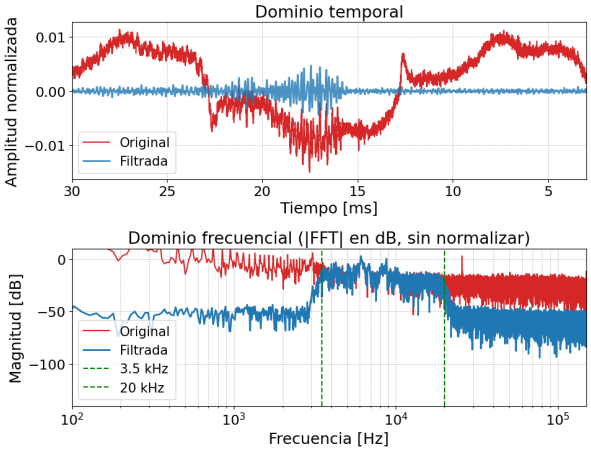


Figura 4: Comparación temporal y espectral de un segmento de señal de cangrejo antes y después del filtrado analógico simulado (Butterworth pasa-banda de orden 10, 3–20 kHz).

V. Datos para el entrenamiento de la red

V-A. *Recolección de los datos para el entrenamiento*

La recolección de datos de individuos de ambas especies de cangrejos se realiza en un entorno controlado de laboratorio, específicamente en la estación J. J. Nágera de la Universidad Nacional de Mar del Plata. En este contexto se registran muestras completas $x[n]$ de dos minutos de duración, muestreadas a una frecuencia $f_s = 300\,\text{kHz}$, utilizando un sistema de adquisición Avisoft UltraSoundGate 116h [14] y un hidrófono Reson TC4013 [15].

El sistema de adquisición utilizado permite registrar un amplio rango de frecuencias, que cubre el espectro completo

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

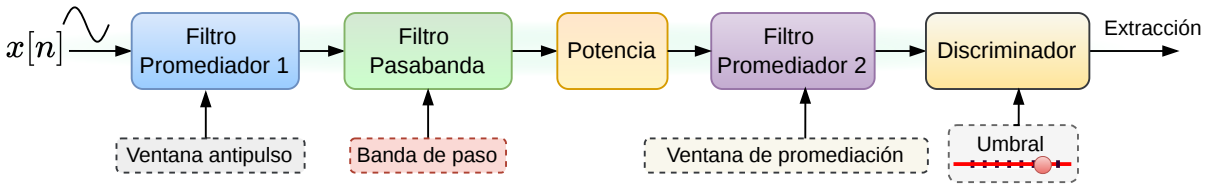


Figura 5: Flujo de procesamiento de señales: esquema de detección automática para la extracción de segmentos acústicos.

de las vocalizaciones de los cangrejos, las cuales pueden extenderse hasta aproximadamente 150 kHz.

Las grabaciones se realizaron en múltiples ensayos independientes, involucrando distintos individuos y condiciones experimentales dentro del entorno controlado. En cada ensayo se registra una única especie, conocida a priori, lo que permite asignar etiquetas confiables a los segmentos detectados dentro de cada archivo. Esto permite conformar un conjunto de datos con variabilidad intraespecífica e interespeccífica, así como diferentes condiciones acústicas de fondo, lo cual resulta relevante para el posterior entrenamiento y evaluación del clasificador.

V-B. Detección de eventos y etiquetado

El esquema de detección automática para la extracción de segmentos etiquetados que se utilizarán en la fase de entrenamiento se ilustra en la Fig. 5. El código utilizado para implementar este esquema puede encontrarse en [16].

1. **Filtro Promediador 1**: promedia una *ventana antipulso* deslizante de 15 puntos de muestra. Este filtro atenúa pulsos unitarios que, al pasar por el filtro pasa-banda, generarían respuestas al impulso no deseadas. La cantidad de muestras de la ventana se calcula de modo de no afectar la banda de interés ($300\ \text{kHz}/20\ \text{kHz} = 15$).
2. **Filtro Pasa Banda**: filtro de muy alto orden con banda de paso $[3\ \text{kHz}, 20\ \text{kHz}]$, que selecciona la región espectral donde se concentran las vocalizaciones de interés.
3. **Cálculo de potencia**: se calcula la potencia instantánea de la señal filtrada.
4. **Filtro Promediador 2**: promedia la potencia instantánea sobre una ventana de 20.000 muestras para estabilizar el piso de ruido y suavizar las variaciones rápidas, facilitando la aplicación de un umbral.
5. **Discriminador**: la potencia promediada se compara con un umbral adaptativo, generando una señal binaria que marca la presencia de eventos acústicos relevantes.

Los intervalos donde el discriminador se activa se utilizan para recortar segmentos de señal. Este esquema permite identificar automáticamente la presencia de eventos acústicos relevantes, separándolos del fondo. Cada segmento se etiqueta a partir de la información experimental asociada a cada grabación: dado que los ensayos se realizan en entornos controlados donde se conoce a priori la especie presente, los segmentos detectados se clasifican como Neo o Cry según corresponda. Por su parte, los intervalos en los que no se detectan eventos se consideran como ruido. De este modo, se genera un conjunto de datos equilibrado que combina tanto vocalizaciones como fragmentos de ruido ambiental. Estos son los datos que se utilizarán para entrenar la red neuronal, en concordancia con la arquitectura prevista para el sistema final (sensor + filtrado + preprocesamiento + red).

En la Fig. 6 se observa parte del proceso de la detección de los segmentos a partir del umbral aplicado sobre la potencia promediada. Dado que los eventos de interés no tienen una duración fija, la ventana temporal que define cada segmento es variable y depende de la activación del discriminador. La figura es una captura en donde se muestra el espectrograma de un segmento del archivo de audio que incluye tres vocalizaciones y la señal resultante de potencia promediada en amplitud.

Para ilustrar la eficiencia que puede aportar un sistema de detección selectiva, en la Tabla I se presenta el porcentaje de información útil por archivo de audio analizado, considerando que la duración promedio de cada evento acústico es de 29.000 muestras.

Tabla I: Resumen de grabaciones y segmentos de interés, incluyendo el porcentaje de información útil. Cada segmento tiene una duración promedio de 29.000 muestras.

Especie	*Neohelice granulata*	*Cyrtograpsus angulatus*
Archivos	805	59
Segm. útiles	4.728	941
% útil	0,95 %	1,29 %
Hs. totales	13,42	1,97
Hs. útiles	0,127	0,025

V-C. Emulación del filtrado

Dado que la arquitectura a embeber incluirá una etapa de filtrado en tiempo real, se aplican por separado los dos filtros propuestos sobre los segmentos previamente recortados y etiquetados. De este modo se generan dos conjuntos de datos independientes, cada uno correspondiente a una alternativa de filtrado, lo que permite emular el efecto que tendría cada opción y evaluar su impacto sobre el desempeño de la red.

V-D. Segmentación y preprocesamiento

A partir de los segmentos previamente detectados y etiquetados, se construyen las instancias de entrada para el

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

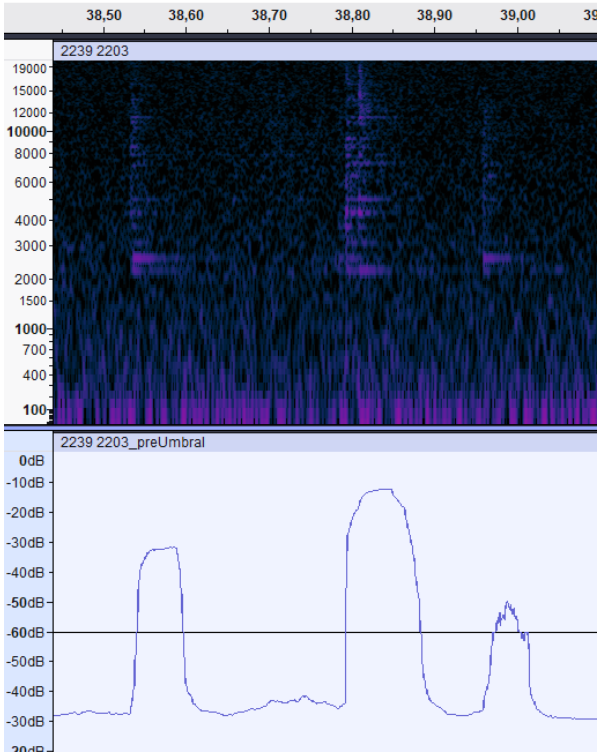


Figura 6: Captura con tres vocalizaciones. Arriba, el espectrograma de una muestra completa y abajo su correspondiente potencia promediada y umbral de comparación. El eje horizontal está en segundos y es compartido por ambas figuras; el eje vertical del espectrograma está en Hz y el de potencia en dB.

entrenamiento del clasificador. Dado que estos segmentos suelen ser más largos de lo necesario para la clasificación, se subdividen en ventanas temporales más cortas. Antes de la segmentación, se aplican estrategias de reducción de datos para disminuir la tasa de muestreo efectiva y la dimensionalidad de las señales, evaluando distintos factores de submuestreo. Sobre cada bloque resultante se realizan operaciones de agregación, como promediado por bloques y cálculo del valor eficaz en ventanas no superpuestas, que permiten reducir la cantidad de datos y resaltar diferentes propiedades de la señal. Finalmente, la señal procesada se segmenta en ventanas de longitud fija, sobre las cuales se extraen las características que alimentan al clasificador. De esta manera es posible analizar combinaciones de tamaño de ventana, submuestreo y preprocesamiento, evaluando su efecto sobre el desempeño del sistema y su viabilidad para identificar especies a partir de segmentos breves, un requisito importante para la implementación en tiempo real sobre hardware embebido.

V-E. Extracción de características

A partir de los segmentos filtrados se extraen características temporales mediante el extractor `catch22` [17], obteniendo vectores de 22 atributos para cada fragmento.

La elección de `catch22` permite generar un vector fijo de 22 características aun cuando los eventos acústicos no poseen una duración constante, lo que resulta especialmente relevante en el caso de los sonidos producidos por los cangrejos. Esta propiedad posibilita obtener una representación compacta y homogénea de cada fragmento sin requerir que la ventana de análisis capture el evento completo ni imponer una estructura temporal uniforme en todos los segmentos. Además, atenúa la sensibilidad al alineamiento exacto del evento dentro de la ventana. Dado que sus descriptores se basan en propiedades estadísticas de corto y mediano plazo, es posible capturar rasgos representativos incluso cuando el evento se encuentra parcialmente contenido o desplazado dentro del segmento. Esto resulta especialmente útil en la implementación en tiempo real, donde el dispositivo procesa ventanas de longitud fija y los eventos acústicos pueden aparecer fragmentados, incompletos o distribuidos de manera irregular.

VI. Clasificador

La construcción del clasificador se aborda como un flujo de trabajo que comprende la preparación de los datos, el entrenamiento del modelo, la validación de su desempeño y la posterior exportación a código en lenguaje C para su ejecución en un microcontrolador. Cabe destacar que esta sección no describe una implementación específica del sistema ni decisiones de diseño a nivel arquitectural, sino el proceso metodológico seguido para el desarrollo del modelo de clasificación.

1. **Preparación del conjunto de datos.** En esta etapa se realizan los ajustes necesarios sobre los datos para el correcto entrenamiento de la red, incluyendo balanceo, normalización y cuantización.
2. **Diseño de la red neuronal.** Se desarrolla una red neuronal cuantizada y de baja complejidad, con el objetivo de minimizar el uso de recursos y permitir su ejecución eficiente en plataformas embebidas de bajo costo. Esta red constituye el modelo *estudiante*.
3. **Entrenamiento mediante destilación del conocimiento.** Se define un modelo neuronal de mayor complejidad y no cuantizado, denominado *maestro*, que se entrena para maximizar el desempeño. Posteriormente, su conocimiento se transfiere al modelo *estudiante* mediante técnicas de destilación.
4. **Preparación para la implementación embebida.** El modelo *estudiante*, ya entrenado y optimizado, se convierte al formato TensorFlow Lite y se cuantiza en `int8`. En esta etapa se evalúan su precisión y sus requerimientos de memoria y cómputo, verificando su compatibilidad con un microcontrolador de bajos recursos.

Estas fases constituyen un proceso iterativo orientado a optimizar el equilibrio entre el uso de recursos y la precisión del modelo. En este contexto, la construcción del conjunto de datos incluye la exploración de distintas combinaciones de tamaño de ventana, factores de submuestreo y estrategias de

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

preprocesamiento, así como la evaluación de filtros digitales y analógicos simulados. Cada una de estas variantes se evalúa en función de su efecto tanto en el rendimiento durante el entrenamiento como en el desempeño final sobre los datos de prueba. En paralelo, en la etapa de diseño de la red se analizan distintas arquitecturas de clasificadores, buscando la mejor relación entre complejidad y exactitud. El flujo de trabajo correspondiente se presenta en la Fig. 7.

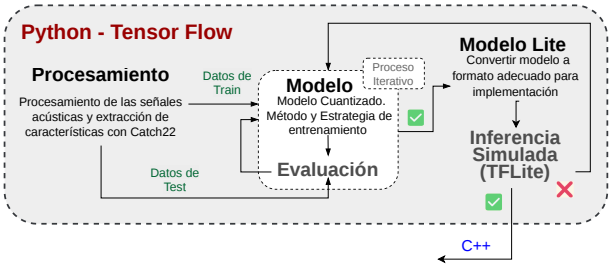


Figura 7: Flujo de trabajo general al que ingresan las señales provenientes del sensor, ya filtradas.

VI-A. Balanceo, normalización y cuantización

Los vectores extraídos se balancean para evitar sesgos entre las tres clases. Para las clases de cangrejos se ajusta el número de vectores de la clase con menor cantidad de datos, mientras que para la clase de ruido se extraen fragmentos de las mismas grabaciones de cangrejos, de manera que todas las clases tengan una representación equivalente.

A partir de estos vectores se generan los conjuntos de entrenamiento (Train), validación (Val) y prueba (Test). La partición del dataset se realiza a nivel de experimento, es decir, la unidad de partición corresponde a cada experimento completo (incluyendo todos sus archivos de audio y los segmentos derivados de los mismos). De esta forma, los experimentos asignados al conjunto de prueba no comparten ningún archivo ni segmento con los utilizados en entrenamiento o validación.

En consecuencia, todos los segmentos pertenecientes a un mismo experimento se asignan exclusivamente a uno de los subconjuntos (Train, Val o Test), evitando la presencia de segmentos correlacionados en distintos conjuntos y descartando posibles fugas de información. Esto garantiza que el conjunto de prueba represente condiciones experimentales completamente no vistas durante el entrenamiento. Además, en el sistema final, las señales de entrada provienen de un convertidor analógico–digital (Analog-to-Digital Converter, ADC) y solo están disponibles como números enteros de 8 bits. Para que el modelo se entrene y evalúe bajo condiciones similares a las del hardware, los datos se normalizan al intervalo [0,1] y se cuantizan uniformemente, reproduciendo así las limitaciones de precisión impuestas por la entrada real del sistema.

VI-B. Diseño de la red neuronal

El modelo de la red seleccionada para su implementación se presenta en la Tabla II. Se trata de una red neuronal cuantizada y de baja complejidad, con un número reducido de capas, que en las etapas posteriores se denomina *estudiante*. Esta baja demanda de recursos asegura su adecuada implementación en hardware de recursos limitados.

Tabla II: Arquitectura de la red *estudiante*. Siendo: kernelQ = quantized_bits(8, 2, α=1), biasQ = quantized_bits(8, 2, α=1), activationQ = quantized_bits(8, 2).

Capa	Salida	# Parámetros
QActivation	(None, 22)	0
fc1 (QDense)	(None, 32)	736
relu1 (QActivation)	(None, 32)	0
fc2 (QDense)	(None, 16)	528
relu2 (QActivation)	(None, 16)	0
fc3 (QDense)	(None, 8)	136
relu3 (QActivation)	(None, 8)	0
output (QDense)	(None, 3)	27
softmax (Activation)	(None, 3)	0
Parámetros totales		**1.427**

VI-C. Entrenamiento

El método de entrenamiento adoptado es la Destilación de Conocimiento [18], la cual permite alcanzar un desempeño elevado mediante el uso de modelos de baja complejidad. En este enfoque, el conocimiento aprendido por un modelo de mayor capacidad (modelo maestro) se transfiere a un modelo más simple (modelo estudiante), favoreciendo una mejor generalización sin incrementar significativamente los requerimientos computacionales. En la Fig. 8 se muestra un esquema de este tipo de entrenamiento.

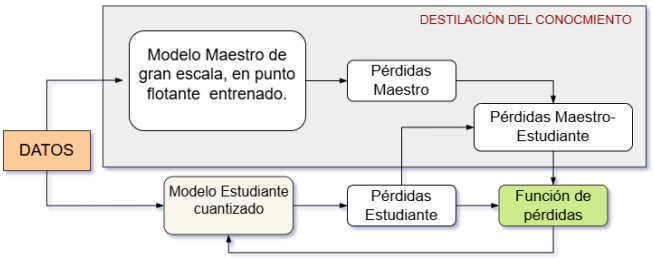


Figura 8: Esquema de entrenamiento por destilación del conocimiento

VI-C1. Elección de red maestro: Con el objetivo de analizar el impacto de distintas arquitecturas de redes neuronales, se evalúan diferentes modelos, incluyendo una red neuronal profunda totalmente conectada, una red convolucional y arquitecturas basadas en *Temporal Convolutional Networks* (TCN) [19]. La selección de estas arquitecturas responde a la comparación entre un modelo clásico, un modelo basado en convoluciones que captura correlaciones locales, y una arquitectura diseñada específicamente para el modelado de secuencias temporales mediante convoluciones causales y dilatadas. En particular, se consideran tanto una TCN estándar como una TCN profunda. Los códigos desarrollados y las características exactas de cada modelo se encuentran disponibles en [20]. El modelo maestro se entrena de manera

elektron
Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026
Original Article

Tabla III: Resumen de arquitecturas de modelos ”Maestro ”propuestos

Modelo	Tipo	Entrada	Componentes principales	Regularización
Modelo 1	MLP (Fully Connected)	Vector (22)	Dense (64-32-16-8) + BN	Dropout (0.2, 0.1), L2
Modelo 2	CNN 1D	Secuencia (22,1)	Conv1D (32-64-64-128) + Pooling + Dense	BatchNorm, Dropout (0.3), L2
Modelo 3	TCN (Residual)	Secuencia (22,1)	Bloques residuales dilatados (1,2,4,8) + Dense	Dropout (0.3)
Modelo 4	TCN + Atención	Secuencia (22,1)	Bloques residuales (1–16) + MultiHeadAttention + GAP	Dropout (0.4), LayerNorm

supervisada utilizando los vectores de características etiquetados correspondientes al conjunto de entrenamiento, mientras que el conjunto de validación se emplea para monitorear el desempeño y seleccionar las configuraciones más adecuadas.

VI-C2. Entrenamiento del Estudiante: Por su parte, el *estudiante* (el modelo cuantizado destinado a la implementación en hardware) aprende a replicar el comportamiento del *maestro*. Es importante destacar que el entrenamiento es consciente de la cuantización (QAT). En este esquema, el entrenamiento del estudiante se realiza utilizando una combinación de la información provista por las etiquetas reales y las salidas del modelo maestro, permitiendo transferir el conocimiento aprendido por este último. De esta manera, el estudiante no solo ajusta sus parámetros para predecir correctamente las clases, sino también para aproximar la distribución de probabilidad generada por el maestro. Los detalles específicos de implementación, incluyendo la configuración de hiperparámetros y las características particulares de cada arquitectura, se encuentran disponibles en [20].

VII. Resultados del entrenamiento

Previo a la presentación de los resultados, se evalúan distintas alternativas para el esquema de filtrado aplicado a las señales; sin embargo, no se observan diferencias significativas en el desempeño de los modelos entrenados bajo ambos enfoques. Dado que el análisis comparativo del impacto del tipo de filtrado no constituye el foco de este trabajo, y con el objetivo de simplificar el esquema de procesamiento, en los experimentos presentados a continuación se utiliza indistintamente uno de los esquemas de filtrado evaluados.

Los resultados obtenidos del entrenamiento de las diferentes arquitecturas del modelo *maestro*, presentadas en la Tabla III, se muestran en la Tabla IV. A partir de estos resultados, se seleccionan los modelos candidatos para su posterior utilización en el proceso de destilación.

Cabe destacar que, durante los experimentos, se emplean distintas estrategias de muestreo y tamaños de ventana. Si bien la frecuencia de muestreo original de las señales es de 300 kHz, la banda de interés se encuentra limitada a frecuencias inferiores a 20 kHz; en consecuencia, se emula un proceso de submuestreo equivalente al uso de un ADC con menor frecuencia de adquisición. Adicionalmente, se aplican distintas estrategias de procesamiento posterior al submuestreo, incluyendo el cálculo de descriptores estadísticos como el valor RMS.

Los resultados presentados en la Tabla IV indican que los candidatos más prometedores a *maestro* son aquellos modelos entrenados con un submuestreo de factor 16 y sin la aplicación de operaciones aritméticas adicionales. Este

Tabla IV: Resultados de entrenamiento para distintas estrategias y modelos del conjunto maestro. Modelos: 1=Red profunda, 2=Red convolucional, 3=Red TCN, 4=Red TCN profunda. Datos: 0,9/0,2/0,1 (Train/Val/Test).

Ventana	Submuestreo / Post-proc.*	Datos	Accuracy	Val Accuracy	Modelo #
1024	16 / —	[2823 x 22]	91 %	90 %	1
1024	16 / —	[2823 x 22]	98 %	89 %	2
1024	16 / —	[2823 x 22]	100 %	88 %	3
1024	16 / —	[2823 x 22]	88 %	88 %	4
1024	2 / RMS4	[6664 x 22]	79 %	78 %	1
1024	2 / RMS4	[6664 x 22]	91 %	78 %	2
1024	2 / RMS4	[6664 x 22]	97 %	75 %	3
1024	2 / RMS4	[6664 x 22]	95 %	75 %	4
1024	4 / RMS4	[2823 x 22]	79 %	78 %	1
1024	4 / RMS4	[2823 x 22]	91 %	78 %	2
1024	4 / RMS4	[2823 x 22]	97 %	75 %	3
1024	4 / RMS4	[2823 x 22]	95 %	75 %	4
1024	8 / RMS2	[2823 x 22]	81 %	78 %	1
1024	8 / RMS2	[2823 x 22]	96 %	80 %	2
1024	8 / RMS2	[2823 x 22]	99 %	82 %	3
1024	8 / RMS2	[2823 x 22]	99 %	81 %	4
512	16 / —	[6664 x 22]	81 %	82 %	1
512	16 / —	[6664 x 22]	85 %	81 %	2
512	16 / —	[6664 x 22]	93 %	81 %	3
512	16 / —	[6664 x 22]	95 %	80 %	4
512	8 / —	[14637 x 22]	76 %	77 %	1
512	8 / —	[14637 x 22]	85 %	76 %	2
512	8 / —	[14637 x 22]	89 %	75 %	3
512	8 / —	[14637 x 22]	89 %	75 %	4
512	4 / —	[30278 x 22]	72 %	72 %	1
512	4 / —	[30278 x 22]	72 %	73 %	2
512	4 / —	[30278 x 22]	85 %	70 %	3
512	4 / —	[30278 x 22]	81 %	70 %	4
512	2 / RMS4	[14637 x 22]			
512	2 / RMS4	[14637 x 22]	55 %	55 %	1
512	2 / RMS4	[14637 x 22]			
512	2 / RMS8	[6664 x 22]			
512	2 / RMS8	[6664 x 22]	79 %	63 %	1
512	2 / RMS8	[6664 x 22]			

Nota: *Post-proc. se refiere a operaciones aplicadas después del submuestreo. RMSX indica cálculo del valor eficaz (RMS) sobre bloques de X muestras.

comportamiento sugiere que la información discriminativa relevante para la tarea de clasificación se preserva aún bajo esquemas de submuestreo agresivos, lo cual podría estar asociado a la concentración del contenido espectral útil en bajas frecuencias. Este aspecto será analizado con mayor detalle en la Sección X. Por otro lado, se observa que las arquitecturas más complejas no aportan mejoras significativas en la performance del modelo maestro. A partir de esta selección, se procede a la destilación del conocimiento hacia el modelo *estudiante*, realizando ajustes finos en parámetros como el tamaño del *batch* y la tasa de destilación, utilizando como maestro el modelo más simple (modelo #1).

En la Fig. 9 se presentan los resultados obtenidos tras el proceso de entrenamiento. La precisión alcanzada por el modelo *maestro* para los datos de validación es de 90 %, mientras que el modelo *estudiante* cuantizado logra una

Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026

precisión para los datos de validación de 83 %. La tasa de aprendizaje se ajusta utilizando la técnica de decaimiento en pasos (*Step Decay*), donde la tasa se reduce de forma escalonada; se emplea un tamaño de batch de 30 y una tasa de destilación de 0,7.

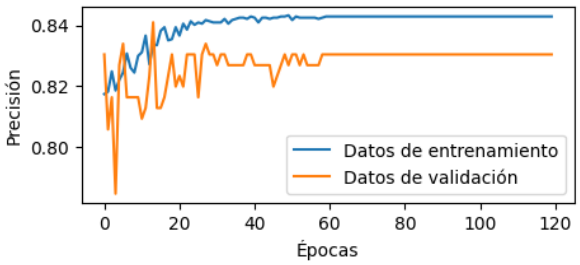


Figura 9: Precisión del modelo cuantizado (*estudiante*) durante el proceso de entrenamiento.

VIII. Implementación

El modelo se entrena utilizando QKeras, una extensión de Keras/TensorFlow que permite simular los efectos de la cuantización mediante capas específicas (QDense, QActivation), y funciones de cuantización como `quantized_bits` [21], [22]. Aunque el entrenamiento se realiza en punto flotante, esta simulación permite que la red aprenda a funcionar bajo las restricciones numéricas que impondría una implementación en baja precisión, anticipando las posibles pérdidas de desempeño. Dado que TensorFlow Lite no es compatible directamente con QKeras, los pesos entrenados deben transferirse a un modelo equivalente implementado en Keras convencional. Este modelo se convierte posteriormente al formato `.tflite`, que constituye el archivo final utilizado para la ejecución del clasificador en el microcontrolador mediante TensorFlow Lite Micro.

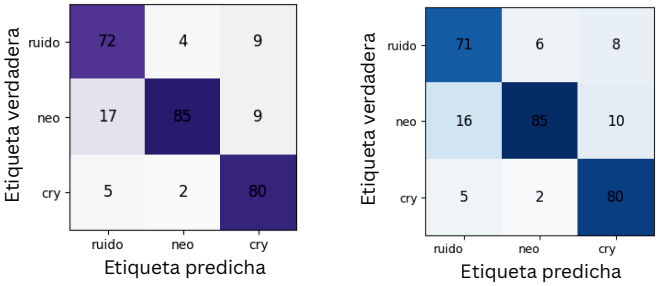
Considerando las limitaciones de las plataformas objetivo, se prioriza la cuantización y el diseño de arquitecturas compactas, no así el pruning, ya que no ofrece beneficios significativos en términos de memoria o latencia en microcontroladores convencionales. En este contexto, se aplica una cuantización entera posterior al entrenamiento (post-training quantization, INT8) utilizando la herramienta oficial de TensorFlow Lite [23]. A diferencia del enfoque simulado de QKeras, esta etapa transforma realmente los pesos y las activaciones a valores enteros de 8 bits, reduciendo tanto el tamaño del modelo como el uso de memoria durante la inferencia. Para garantizar una cuantización precisa, se utiliza un conjunto representativo de datos que permite calibrar automáticamente los parámetros de escala y punto cero de cada tensor.

Durante la inferencia, el modelo espera recibir entradas escaladas al mismo rango utilizado durante la calibración. Dado que los datos reales provienen de un ADC y se representan como enteros sin signo de 8 bits, se incorpora una etapa de escalado que ajusta estos valores al rango requerido por el modelo en `int8`. Los parámetros necesarios para este escalado son provistos automáticamente por la herramienta de conversión a TensorFlow Lite, lo cual asegura la coherencia entre las etapas de entrenamiento, cuantización y ejecución en hardware.

La salida del modelo es una predicción de clase, que indica si el fragmento de audio corresponde a *Neohelice granulata*, *Cyrtograpsus angulatus* o ruido ambiental.

IX. Resultados obtenidos

En la Fig. 10 se muestran las matrices de confusión correspondientes al modelo cuantizado y al modelo convertido al formato TFLite. Estas matrices se obtienen evaluando ambos modelos sobre el conjunto de datos de prueba, separado previamente para garantizar que no hayan tenido acceso a estos datos durante el entrenamiento. Las precisiones alcanzadas sobre el conjunto de prueba son del 83,75 % tanto para el modelo cuantizado como para el modelo implementado mediante TFLite.



(a) Modelo cuantizado. (b) Modelo final.

Figura 10: Comparación de las matrices de confusión obtenidas para el modelo cuantizado y para la simulación de su implementación sobre un microcontrolador.

La inferencia del modelo final se realiza utilizando la API *Interpreter* de TensorFlow Lite. Esta API permite ejecutar el modelo TFLite en un entorno de Python, emulando las condiciones de ejecución que luego se tendrán en el microcontrolador. En este caso particular, el modelo emplea cuantización int8, con escala 0,0039215 y punto cero -128, mapeando el rango real al intervalo entero $[-128, 127]$.Bajo esta configuración, el modelo presenta una ocupación de 1,41 kB de memoria RAM asociada a los tensores durante la inferencia, mientras que el tamaño del modelo almacenado es de 4,15 kB.

X. Análisis de los Resultados Obtenidos

Determinar el impacto de cada bloque del sistema sobre el uso de recursos y el desempeño global en una plataforma embebida de bajo costo es de suma relevancia para la implemetación. La Tabla V resume una estimación del consumo de memoria y cómputo asociado a cada uno de los bloques principales del sistema. A partir de estos valores se establece el peso relativo de cada etapa dentro de la arquitectura propuesta y se identifican los componentes más exigentes en términos de recursos.

Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026

Tabla V: Estimación de recursos requeridos por cada bloque sobre un microcontrolador con 520 kB RAM y 4 MB Flash.

Bloque	Flash [kB]	RAM [kB]	Uso de CPU	Porcentaje del recurso
Filtro Butterworth 6º orden (3 SOS)	–	0.10	3–5 %	<0.01 % Flash, 0.02 % RAM
Extracción de características (`catch22`)	≈15	≈5	15–25 %	1.0 % Flash, 1.0 % RAM
Clasificador TFLite (`int8`)	4.46	1.48	<1 %	0.1 % Flash, 0.3 % RAM

Las estimaciones reportadas en la Tabla V se obtuvieron mediante una combinación de herramientas de simulación y análisis directo de implementación. En el caso del clasificador, el consumo de memoria (Flash y RAM) se obtuvo a partir de la herramienta TensorFlow Lite, que proporciona automáticamente el tamaño del modelo y la asignación de tensores durante la inferencia. Para la etapa de extracción de características (catch22), los valores se estimaron a partir de su implementación en C, considerando tanto el tamaño del código compilado como la memoria necesaria para buffers intermedios. En el caso del filtrado digital, los requerimientos de memoria y cómputo se estimaron analíticamente a partir de la estructura del filtro IIR, teniendo en cuenta la cantidad de secciones de segundo orden y las variables de estado asociadas. El uso de CPU se aproximó en función del número de operaciones por muestra y fue contrastado mediante perfiles de ejecución en plataformas embebidas representativas. La referencia a un microcontrolador con 520 kB de RAM y 4 MB de Flash corresponde a una configuración típica dentro de la familia STM32 (por ejemplo, dispositivos de la serie STM32F4), comparable con plataformas ampliamente utilizadas como ESP32. El objetivo de esta referencia es validar la viabilidad del sistema en hardware de bajo costo, más que apuntar a un dispositivo específico. En este sentido, los valores reportados deben interpretarse como estimaciones de orden de magnitud orientadas al diseño del sistema.

En la Tabla V se observa que la etapa de extracción de características mediante `catch22` constituye el bloque más costoso del sistema en términos de memoria y uso de CPU. Esta etapa requiere aproximadamente 15 kB de Flash, 5 kB de RAM y entre un 15 % y un 25 % del tiempo de procesamiento, superando ampliamente a los requerimientos del filtrado y del clasificador cuantizado.

En contraste, el filtrado digital presenta un impacto mínimo sobre los recursos del sistema. El filtro Butterworth pasabanda de sexto orden requiere del orden de 100 bytes de memoria y un uso de CPU inferior al 5 %, lo que confirma que su implementación directa en el microcontrolador resulta viable y no constituye un factor limitante. En términos de desempeño del clasificador, no se observan diferencias significativas entre el uso de filtrado digital y analógico, tal como se evidenció en las simulaciones comparativas realizadas. Por otra parte, dado que el filtrado digital presenta un impacto mínimo en el uso de recursos del microcontrolador (Tabla IV), su implementación resulta suficiente dentro del esquema propuesto.

Asimismo, el análisis presentado en esta sección respecto del filtrado se centra exclusivamente en su impacto en los recursos del sistema embebido y en la viabilidad de su implementación, y no en una comparación exhaustiva entre distintas técnicas de filtrado. En este sentido, las consideraciones realizadas tienen como objetivo justificar la elección de una solución compatible con las restricciones de hardware sin afectar el desempeño del clasificador.

En relación con el submuestreo considerado durante la etapa de entrenamiento, se observa que una tasa efectiva equivalente a 300 kHz/16 resulta compatible con las prestaciones de un microcontrolador de bajo costo, cumpliendo con las restricciones de hardware de recursos limitados. Si bien esta frecuencia no satisface estrictamente el criterio de Nyquist para toda la banda de interés (0–20 kHz), el submuestreo con factor 16 proporciona el mejor desempeño del clasificador en las pruebas realizadas. Cabe señalar que el criterio de Nyquist garantiza la reconstrucción fiel de una señal original, pero no constituye un requisito estricto en tareas de clasificación, donde el objetivo es preservar información discriminativa y no la forma exacta de la señal. En este contexto, el submuestreo con factores elevados introduce aliasing; sin embargo, dicha transformación no implica necesariamente la pérdida de las características relevantes para distinguir entre clases. Esto sugiere que la información discriminativa de los eventos acústicos de interés se encuentra concentrada en componentes espectrales que se preservan incluso bajo un submuestreo agresivo, permitiendo una representación suficiente para la tarea de clasificación. En este esquema, el almacenamiento del dato acústico completo puede realizarse mediante un sistema de adquisición externo de mayor velocidad, desacoplando la etapa de clasificación en tiempo real del proceso de registro de alta resolución.

Respecto del diseño del modelo maestro utilizado para la destilación del conocimiento, se observa que arquitecturas de mayor complejidad logran un desempeño ligeramente superior sobre los datos de entrenamiento, pero no producen mejoras significativas en los resultados de validación. Este comportamiento indica que el aumento de complejidad no se traduce en una mejor capacidad de generalización y sugiere la existencia de un límite impuesto por la calidad y la naturaleza del conjunto de datos disponible.

En relación con la composición del conjunto de datos, se observa una menor disponibilidad de registros correspondientes a la especie *Cyrtograpsus angulatus*, lo que se traduce en una menor cantidad de eventos útiles respecto de otras clases. Aunque el balanceo asegura que la cantidad de eventos sea igualitaria.

Finalmente, si bien la red neuronal representa conceptualmente la etapa más compleja del sistema, su versión cuantizada presenta una carga computacional muy reducida. El clasificador final cuenta con 1.427 parámetros y requiere menos del 1 % de uso de CPU, lo que lo vuelve prácticamente despreciable desde el punto de vista computacional. En

Revista Elektron, Vol. 10, No. 1, pp. 41–51 (2026)
https://doi.org/10.37537/rev.elektron.10.1.227.2026

consecuencia, se establece que el consumo de recursos del sistema se encuentra dominado por la extracción de características, mientras que el filtrado digital y la clasificación no introducen restricciones relevantes para la operación en tiempo real.

XI. Conclusiones y Trabajo Futuro

Los resultados obtenidos muestran que el modelo desarrollado presenta un uso de recursos altamente eficiente, resultando adecuado para su integración en un sistema embebido de bajo costo. El entrenamiento realizado a partir de señales adquiridas con el sensor real, posteriormente filtradas y segmentadas en las tres clases de interés (Neo, Cry y ruido), garantiza una adecuada correspondencia entre las condiciones de laboratorio y el escenario operativo previsto.

La exactitud promedio alcanzada del 83,75 % resulta adecuada para la aplicación propuesta, cuyo objetivo principal es reducir la carga de grabación en campo mediante la detección selectiva de eventos acústicos relevantes. En este contexto, el sistema actúa como un filtro inteligente previo al análisis posterior por parte de un operador humano.

Asimismo, el análisis de los archivos de audio evidencia que solo el 1,12 % del total de los datos contiene información relevante, lo que resalta la ventaja de implementar un clasificador embebido capaz de detectar y almacenar únicamente dichos segmentos.

Un aspecto relevante de la implementación embebida es que el clasificador puede operar con una frecuencia de muestreo efectiva 16 veces menor y con ventanas de tan solo 1024 muestras, lo que lo hace totalmente compatible con microcontroladores de muy bajos recursos. Esto permite realizar la detección de eventos de cangrejo de manera eficiente sin comprometer la capacidad de memoria del sistema principal. La parte más crítica recae en la gestión de memoria externa, donde se utiliza un buffer de mayor tamaño (aproximadamente 16.500 muestras) para almacenar temporalmente los eventos completos antes de transferirlos a memoria externa. Este enfoque asegura que los eventos completos se preserven para su análisis posterior, mientras que la etapa de clasificación se mantiene altamente eficiente y liviana en términos de recursos computacionales.

En esta misma línea, se plantea la ampliación del conjunto de datos, especialmente en lo que respecta a la especie *Cyrtograpsus angulatus*, con el objetivo de incrementar la variabilidad intraclase disponible y mejorar la evaluación de la capacidad de generalización del modelo.

Como trabajo futuro, se prevé ampliar la evaluación del modelo incorporando métricas adicionales más allá de la matriz de confusión. En particular, resulta de interés el costo esperado normalizado (CEN), una de las pocas métricas consistentes en problemas de clasificación arbitrarios. No obstante, su aplicación requiere la definición de costos asociados a los errores en un escenario real, por lo que se abordará en etapas posteriores de validación en campo.

Paralelamente, se avanzará en la implementación integral del sistema en el dispositivo embebido, incluyendo la adquisición en tiempo real y su validación experimental en condiciones reales de operación.

Finalmente, se identificó que la etapa de extracción de características mediante `catch22` constituye el componente más costoso en términos de recursos. En consecuencia, se explorarán alternativas más livianas basadas en estadísticas simples o transformaciones optimizadas para hardware, con el objetivo de reducir el uso de memoria y cómputo sin degradar significativamente el desempeño del sistema.

Declaración de disponibilidad de datos

Los datos que respaldan los resultados de este estudio están disponibles a partir del autor de correspondencia previa solicitud razonable.

CRediT declaración de contribución de autoría

M. Celeste Cebedio: Validación; redacción – borrador original; Adquisición de fondos. **Martín Lorusso:** Validación; Investigación. **Leonardo Arnone:** Software; visualización. **Lucas A. Rabioglio:** Software; redacción; Conceptualización. **Maximiliano Antonelli:** investigación; visualización; Software; redacción. **Raúl E. Lopresti:** visualización; redacción. **Luciana De Micco:** redacción – borrador original; investigación; Validación; supervisión; Análisis formal. **M. Paz Sal Moyano:** administración del proyecto; Adquisición de fondos; Investigación.

XII. Agradecimientos

Este trabajo fue financiado por la Universidad Nacional de Mar del Plata a través de los proyectos EXA1213/24 “Bioacústica marina: señales sonoras naturales y efecto del sonido antrópico en una especie clave de cangrejo. OCS 2024-6” y PI2Ba RR-2024-1914 “Técnicas avanzadas de ingeniería e inteligencia artificial aplicadas al análisis de datos acústicos biológicos: caracterización a nivel especies e intra-especie y efecto de la contaminación por ruido sobre animales”.

Referencias

[1] M. Minello, L. Calado y F. C. Xavier, “Ecoacoustic Indices in Marine Ecosystems: A Review on Recent Developments, Challenges, and Future Directions”, *ICES Journal of Marine Science*, vol. 78, n.º 9, págs. 3066-3074, 2021. DOI: 10.1093/icesjms/fsab193. dirección: https://academic.oup.com/icesjms/article-pdf/78/9/3066/41765284/fsab193.pdf.

[2] Ministerio de Ambiente y Desarrollo Sostenible de la Nación Argentina, *Ficha técnica de la Reserva de la Biosfera Mar Chiquita*, 2023. dirección: https://www.argentina.gob.ar/sites/default/files/2023/02/fichas_web_07.pdf.

[3] AMPAR Argentina, *Mar Chiquita*, Accessed: Apr. 26, 2025. dirección: https://ampargentina.org/areas/mar-chiquita/.

[4] M. P. Sal Moyano, M. Ceraulo, T. Luppi, M. A. Gavio y G. Buscaino, "Anthropogenic and Biological Sound Effects on the Maternal Care Behavior of a Key Crab Species", *Frontiers in Marine Science*, vol. 10, 2023. DOI: 10.3389/fmars.2023.1050148. dirección: https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2023.1050148.

[5] S. Kumar et al., *Deep Learning in Computational Biology: Advancements, Challenges, and Future Outlook*, 2023. arXiv: 2310.03086 [cs.LG]. dirección: https://arxiv.org/abs/2310.03086.

[6] D. Tuia et al., "Perspectives in Machine Learning for Wildlife Conservation", *Nature Communications*, vol. 13, n.º 792, 2022. DOI: 10.1038/s41467-022-27980-y.

[7] A. Lamba, P. Cassey, R. Raja Segaran y L. Koh, "Deep Learning for Environmental Conservation", *Current Biology*, vol. 29, n.º 19, R977-R982, 2019. DOI: 10.1016/j.cub.2019.08.016.

[8] U. Malik, M. Malik y A. Malik, "Leveraging Deep Learning for Accurate Classification of Leptograpsus Crabs Based on Morphological Measurements", en *Intelligent Computing Systems*, A. Safi, A. Martin-Gonzalez, C. Brito-Loeza y V. Castañeda-Zeman, eds., Cham: Springer Nature Switzerland, 2025, págs. 161-175, ISBN: 978-3-031-82931-4. DOI: 10.1007/978-3-031-82931-4_12.

[9] C. Wu et al., "A Part-based Deep Learning Network for Identifying Individual Crabs Using Abdomen Images", *Frontiers in Marine Science*, vol. 10, 2023. DOI: 10.3389/fmars.2023.1093542. dirección: https://www.frontiersin.org/journals/marine-science/articles/10.3389/fmars.2023.1093542.

[10] Espressif Systems, *ESP32 Series Datasheet*, 2024. dirección: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[11] STMicroelectronics, *Artificial Intelligence on STM32 Microcontrollers*, 2024. dirección: https://www.st.com/en/embedded-software/x-cube-ai.html.

[12] STMicroelectronics, *STM32 Microcontrollers*, 2024. dirección: https://www.st.com/stm32.

[13] V. V, A. C. Ramachandra, R. Prasanna, P. C. Kakarla, V. P. J. Simha y N. Mohan, *Implementation of Tiny Machine Learning Models on Arduino 33 BLE for Gesture and Speech Recognition*, 2022. DOI: 10.48550/arXiv.2207.12866. arXiv: 2207.12866 [eess.AS]. dirección: https://arxiv.org/abs/2207.12866.

[14] Avisoft Bioacoustics, *UltraSoundGate 116H: USB-based Ultrasound Recording Interface*, Datasheet, 2023. dirección: https://avisoft.com/ultrasoundgate/116h/.

[15] Teledyne Marine / RESON, *TC4013 Miniature Reference Hydrophone Datasheet*, Product leaflet / technical specifications, 2020. dirección: https://teramara.ca/sites/default/files/2022-01/reson-TC4013%20product%20leaflet.pdf.

[16] M. Antonelli, *scrubDetection*, Accessed: Nov. 18, 2025, 2025. dirección: https://github.com/maxanto/scrubDetection.

[17] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher y N. S. Jones, "catch22: CAnonical Time-series CHaracteristics: Selected through Highly Comparative Time-Series Analysis", *Data Mining and Knowledge Discovery*, vol. 33, n.º 6, págs. 1821-1852, 2019. DOI: 10.1007/s10618-019-00647-x.

[18] J. Gou, B. Yu, S. J. Maybank y D. Tao, "Knowledge Distillation: A Survey", *International Journal of Computer Vision*, vol. 129, n.º 6, págs. 1789-1819, 2021. DOI: 10.1007/s11263-021-01453-z.

[19] S. Bai, J. Z. Kolter y V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", en *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. DOI: 10.48550/arXiv.1803.01271.

[20] M. C. Cebedio, *Clasificación de Cangrejos*, Repositorio de código asociado al clasificador, 2025. dirección: https://github.com/cebedio/Clasificaci-n-de-cangrejos.

[21] M. C. Cebedio, L. A. Rabioglio y L. De Micco, "Quantized Generative Autoencoder for Audio Spectrograms", *IEEE Embedded Systems Letters*, págs. 419-422, jun. de 2025. DOI: 10.1109/LES.2025.3575372.

[22] Google, *QKeras: Quantization Extensions for Keras*, Accessed: Feb. 1, 2025, 2023. dirección: https://github.com/google/qkeras.

[23] TensorFlow, *TensorFlow Lite for Microcontrollers*, Accessed: Apr. 27, 2025, 2025. dirección: https://www.tensorflow.org/lite/microcontrollers?hl=es-419.