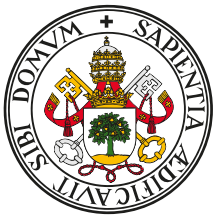


Universidad de Valladolid

Escuela Técnica Superior de
Ingenieros de Telecomunicación

Análisis del uso de algoritmos de Deep Learning en un Sistema acústico de detección de larvas de insectos xilófagos

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías
Específicas de Telecomunicación.
Mención Sistemas de Telecomunicación

Autor:
Alberto García García
Tutora:
Lara del Val Puente

Valladolid, julio 2024

Título	**Análisis del uso de algoritmos de Deep Learning en un Sistema acústico de detección de larvas de insectos xilófagos**
Autor	**D. Alberto García García**
Tutora	**Dra. Lara del Val Puente**
Departamento	**Teoría de la Señal y Comunicaciones**

Tribunal

Presidente	**Alberto Izquierdo Fuente**
Secretaria	**Lara del Val Puente**
Vocal	**Juan José Villacorta Calvo**
Suplente 1	**Ramón de la Rosa Steinz**
Suplente 2	**Alonso Alonso Alonso**

Fecha	**10 de julio de 2024**
Calificación	

A mi familia, por darme la oportunidad
y acompañarme en este camino.

A todo el equipo del GPA, en especial a
mi tutora Lara, por permitirme aprender
y disfrutar tanto de este proyecto.

A todos los amigos que me llevo de la Escuela,
por hacer de esta una etapa inolvidable.

A mi pareja, por su apoyo incondicional,
su paciencia y por creer siempre en mí.

Resumen

Este Trabajo Fin de Grado se centra en la investigación sobre la detección de insectos xilófagos mediante técnicas acústicas, un área ampliamente estudiada pero con ciertas limitaciones que este trabajo busca reducir. Para ello, se han utilizado imágenes acústicas capturadas por un array de micrófonos MEMS, a partir de las cuales se estima la localización de larvas basándose en la energía de la emisión acústica producida por su actividad vital, principalmente su alimentación a base de madera.

Se cuenta con una base de datos de 148,000 señales capturadas a partir de seis larvas de *Hylotrupes bajulus*, colocadas en cuatro vigas de madera diferentes a una distancia de 60 centímetros del array acústico, dentro de una cámara anecoica. Para validar las detecciones realizadas, se han desarrollado varios algoritmos de aprendizaje profundo con el objetivo de reducir las detecciones de falsos positivos, que actualmente se producen debido a la energía radiada por zonas de la madera distintas al lugar donde se originó la emisión acústica, resultando en una localización errónea.

Se obtuvieron resultados prometedores, con una precisión del 98.8 % en el mejor modelo y un valor de área bajo la curva ROC de 0.998. El problema de clasificación se centra en discriminar entre las detecciones capturadas que provienen directamente del lugar de origen de la emisión acústica y aquellas que han atravesado la madera y han sido radiadas a través de otra zona, como un nudo.

Palabras clave

Detección de insectos xilófagos, array acústico, sensores MEMS, redes neuronales, aprendizaje profundo, aprendizaje transferido, redes neuronales convolucionales, perceptrón multicapa.

Abstract

This thesis focuses on research into the detection of wood-boring insects using acoustic techniques, an extensively studied area with certain limitations that this work aims to mitigate. To achieve this, acoustic images captured by an array of MEMS microphones have been utilized to estimate the location of larvae based on the energy of acoustic emissions produced by their vital activity, primarily their wood-feeding behavior.

The study includes a database of 148,000 signals collected from six larvae of *Hylotrupes bajulus*, placed on four different wooden beams at a distance of 60 centimeters from the acoustic array, within an anechoic chamber. To validate the detections made, several deep learning algorithms have been developed with the goal of reducing false positives, which currently occur due to energy radiated from areas of the wood other than the origin of the acoustic emission, resulting in incorrect localization.

Promising results were obtained, with an accuracy of 98.8 % in the best model and an area under the ROC curve value of 0.998. The classification challenge focuses on distinguishing between captured detections originating directly from the site of acoustic emission and those that have passed through the wood and have been radiated through another area, such as a knot.

Keywords

Detection of xylophagous insects, acoustic array, MEMS sensors, neural networks, deep learning, transfer learning, convolutional neural netoworks, multi-layer perceptron.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Estado del arte

La degradación de estructuras de madera por el ataque de insectos xilófagos es un problema global, parcialmente inevitable, que en ciertas ocasiones causa enormes pérdidas económicas [1] y medioambientales [2]. Además representan una amenaza importante para los Bienes de Interés Cultural (BIC) realizados sobre materiales orgánicos como cuadros, esculturas, muebles, etc. [3] En este caso, es crítica la detección prematura para evitar daños antes de que la pieza se encuentre en un estado avanzado de infestación, lo que precisa de técnicas de detección no invasivas que no pongan en riesgo la salud de la obra.

A lo largo del tiempo se han desarrollado diversas técnicas para la detección de estos insectos. Una de las primeras y más sencillas es la inspección visual, esta vía de detección se basa en la observación directa de los agujeros socavados por las larvas, así como cualquier otro indicio visible de la presencia de insectos. A pesar de ser una técnica rudimentaria, actualmente sigue siendo usada por sus ventajas, entre las que destaca su bajo coste, puesto que no se requiere de ningún equipo especializado. No obstante, esta técnica es muy limitada y depende de la experiencia de la persona que realice la inspección, por lo que en ocasiones el insecto puede no ser detectado.

Es por eso que es necesario recurrir a métodos más sofisticados que permitan una detección automática y fiable. Entre las distintas opciones existentes, las más habituales son: la detección acústica, óptica, las técnicas de escaneo y radiografía, el análisis químico y el análisis de ADN (ácido desoxirribonucleico) ambiental.

La detección acústica es un método ampliamente usado para la detección de insectos xilófagos, aprovechando las emisiones acústicas producidas por la actividad de estos. Existe una extensa literatura que aborda este tema y la instrumentación empleada para lograr la detección es diversa.

A pesar de que varios estudios emplean sensores piezoeléctricos [4, 5, 6] o acelerómetros [7, 8, 9, 10, 11] para capturar las señales con el objetivo de reducir las detecciones de ruido exterior, la necesidad de estos dispositivos de estar en contacto directo con la superficie de la madera puede no ser adecuada en algunas ocasiones. Es por esto, que un array acústico se presenta como una alternativa para la detección a distancia. Aunque existe la posibilidad de que pueda captar ruidos externos, la conformación realizada posteriormente permite

reducir considerablemente estos ruidos, filtrando espacialmente las señales adquiridas.

Estudios como [7] han empleado técnicas de Inteligencia Artificial para discriminar entre los sonidos realizados por una larva, de muy baja amplitud, y el ruido de fondo, consiguiendo resultados de precisión cercanos al 90 % entrenando un algoritmo SVM (*Support Vector Machine*). Por su parte, Liu *et al.* proponen en [12] diversas arquitecturas de redes neuronales para reducir el ruido de las señales captadas por un micrófono, tanto con la señal en el tiempo como con su espectro en frecuencia.

En el ámbito de la detección acústica existen diversas formas de abordar el problema, se puede hacer un análisis en el dominio del tiempo, también se puede estudiar el dominio frecuencial, o incluso aprovechar una representación tiempo-frecuencia. En [4] se hace un análisis de las señales en el domino temporal que permite la detección de mordidas producidas por larvas de *Hylotrupes bajulus* con transductores piezoeléctricos mediante el cálculo de parámetros basados en la dimensión fractal.

Un factor relevante en la detección acústica de larvas de insectos, es la diferencia entre los sonidos producidos en función del tamaño de la larva. En [5] se estudia la dependencia de la energía de la emisión acústica en función del tamaño de la larva y de la distancia a la que se ubica el sensor.

Otra forma de detección es a través de sensores ópticos, estudiada en diversos artículos [13, 14]. Algunas de las técnicas empleadas para este fin requieren de equipos con un coste muy elevado, lo que hace que sea inviable su producción a gran escala para monitorizar periódicamente estructuras de interés. Es por esto, que en [14] se presenta el desarrollo de un sensor electrónico de bajo consumo para incrustar en la madera. Entre varias técnicas investigadas y tras los experimentos preliminares necesarios, los autores proponen la aplicación de técnicas de detección óptica basadas en variaciones en la cantidad de luz absorbida/reflejada. La detección de insectos se realiza utilizando LEDs de iluminación y sensores de luz que registran variaciones en la reflexión cuando un insecto como termitas, hormigas, cucarachas, etc., se acerca al dispositivo. A pesar de las ventajas que esta técnica presenta, como la monitorización periódica a un bajo coste, es necesario incrustar el sensor en la madera convirtiéndose así en un método invasivo.

Por su parte, en [13] se presenta una técnica similar, que incorpora un sistema de alarma por SMS (del inglés, *Short Message Service*) basado en un módulo GSM (del inglés, *Global System for Mobile Communications*) que permitió la monitorización continua de tres edificios.

Otra rama de estudio aplicada a la detección de insectos xilófagos son las imágenes por resonancia magnética (MRI), analizada por Nicosia *et al.* en [3]. Esta técnica se basa en las propiedades magnéticas del núcleo de los átomos y permite obtener una imagen bidimensional (2D) o tridimensional (3D) del objeto bajo estudio. Se obtuvieron resultados positivos en la detección de larvas de especies como *Anisakis simplex* o *Hylotrupes bajulus*, sin embargo, esta técnica requiere de un equipo especializado y poco portable.

En este contexto, este trabajo se presenta como una herramienta de ayuda a la mejora de la precisión de los sistemas acústicos de detección de estos insectos, haciendo de ellos una solución ideal para el control y detección de plagas, así como para la conservación del patrimonio artístico y cultural.

1.2. Objetivos

1.2.1. Objetivos generales

En este trabajo, se pretende desarrollar un algoritmo de *Deep Learning* (DL) que mejore la precisión en la localización de ejemplares de larva del escarabajo de la carcoma grande (*Hylotrupes bajulus*). El objetivo principal es diseñar y evaluar un algoritmo capaz de discriminar si la localización de la larva proporcionada por el array acústico corresponde efectivamente con el punto donde se originó el sonido, o si dicha detección es resultado de la energía radiada a través de un nudo de la madera cercano al lugar donde la larva mordió.

1.2.2. Objetivos específicos

Para conseguir el objetivo general, ha sido necesaria la consecución de los siguientes objetivos específicos.

- Realizar un análisis exhaustivo de los datos en busca de potenciales patrones que distinguieran ambos grupos, con el objetivo de encontrar la mejor forma de entrada a la red neuronal.
- Implementar un algoritmo de segmentación multiblanco en LabVIEW que permita extraer el mayor número de detecciones posible.
- Etiquetar convenientemente los datos para crear una base de datos que sirva para el entrenamiento de los distintos modelos y para el análisis del comportamiento de las larvas. Para esto es necesario definir qué regiones del plano se consideran pertenecientes a una larva, a un nudo, o si se tratan de ruido.
- Estudio de la evolución temporal de las larvas, con el objetivo de determinar si existe o no correlación entre la aparición de un "sonido de nudo" y uno de larva.
- Búsqueda y cálculo de distintas representaciones de las señales, como escalograma, espectrograma y espectro en frecuencia.
- Exploración de distintos algoritmos ya existentes sobre clasificación automática mediante señales acústicas con el fin de encontrar un modelo con el que hacer *transfer learning*.
- Búsqueda y desarrollo de algoritmos de Inteligencia Artificial (IA) para resolver el problema de clasificación.
- Evaluación y comparación de los distintos modelos con diferentes tipos de datos.

1.3. Metodología

Para el desarrollo de este proyecto, esta memoria se desarrolla en las siguientes partes. Primero se expone un marco teórico que aborda todos los fundamentos necesarios para la comprensión del resto del documento. Una vez explicados todos estos conceptos, la

estructura seguida es la que se llevó a cabo durante la experimentación, comenzando por el procesamiento de las señales capturadas, continuando por el desarrollo de distintos algoritmos y por último evaluando los resultados. A continuación se detallan dichas fases:

1. **Adquisición y procesado de las señales**. En el Capítulo 3 se detalla el procesamiento seguido para obtener los distintos datos de entrada para las posteriores redes neuronales. Se evalúa la naturaleza de las señales "en bruto" para calcular parámetros útiles para la clasificación, como serán la Transformada Wavelet o la Transformada Discreta de Fourier (DFT).

2. **Desarrollo de los distintos algoritmos**. El Capítulo 4 aborda la implementación de distintas arquitecturas de redes neuronales profundas utilizadas para la consecución del objetivo del trabajo.

3. **Evaluación de los resultados y conclusiones**. Los Capítulos 5 y 6 analizan los resultados obtenidos y se extraen las conclusiones pertinentes.

Capítulo 2

Marco teórico

En este capítulo se detallan los fundamentos teóricos asociados a este proyecto. Se explicará el funcionamiento y técnicas asociadas a los arrays acústicos, así cómo la generación del sonido por parte de insectos xilófagos y la madera que los alberga, pasando por los conceptos asociados a las redes neuronales y finalizando con una breve explicación de las herramientas de desarrollo empleadas.

2.1. Arrays de micrófonos

Un array no es más que una agrupación de sensores distribuidos en el espacio. En el caso de los arrays acústicos estas agrupaciones son de micrófonos. Las aplicaciones de los arrays son muy variadas y están extendidas en campos muy diversos como SONAR (del inglés *SOund Navigation And Ranging*), Radioastronomía, Comunicaciones, etc. [15]. Sin embargo, en esta sección nos centraremos únicamente en arrays acústicos.

Las geometrías y distribuciones en las que se puede diseñar un array son muy variadas, pero pueden clasificarse atendiendo a su geometría como lineales (1D), planares (2D) o volumétricos (3D) y según la distribución espacial entre sensores como uniformes, no uniformes (ej. logarítmicos) y aleatorios.

Con estos dispositivos es posible muestrear una señal en el dominio espacial. Para poder obtener esta información espacial, es necesario procesar adecuadamente y de manera conjunta las señales recibidas por todos los sensores. Este procesado se conoce como conformación o *beamforming*.

2.1.1. El conformador de haz

El término *beamforming* proviene de los primeros filtros espaciales, diseñados para formar haces (*beams* en inglés) afilados con el objetivo de recibir una señal radiando desde una posición específica y atenuando las señales provenientes de otras localizaciones [16].

Su principio de operación se basa en procesar conjuntamente las señales recibidas en un array por todos los sensores, con el objetivo de conseguir estimar la posición de la señal recibida y el ángulo de llegada de la onda respecto al array, en caso de que se cumpla la

condición de campo lejano y por tanto se pueda asumir una onda plana. Si el array es unidimensional, podremos discriminar entre posiciones a lo largo de un plano, mientras que con un array bidimensional es posible distinguir entre dos componentes de la posición (azimut y elevación) y por último, con un array tridimensional es posible la localización en las tres dimensiones.

Diagrama de radiación

El conformado se comporta como un filtro espacial para cada uno de sus haces, de forma que para el ángulo de llegada deseado deja pasar la señal y para el resto de ángulos se atenúa. A través del diagrama de radiación de un array se puede caracterizar este comportamiento, aunque una de sus ventajas es la posibilidad de cambiar este diagrama electrónicamente en un tiempo muy corto, lo que permite hacer un barrido del espacio electrónicamente.

El diagrama de radiación de un array con separación uniforme entre elementos se define en [15] según la siguiente expresión

$$F(\psi) = \frac{1}{N}\frac{\sin(N\frac{\psi}{2})}{\sin\frac{\psi}{2}}, \quad -\infty < \psi < \infty \tag{2.1}$$

donde

- N es el número de sensores que conforman el array.
- $\psi = kd\cos\theta + \alpha$.
- k es el número de onda y se define como $2\pi/\lambda$, siendo λ la longitud de onda de la señal.
- d es la distancia entre los sensores del array.
- θ es el ángulo de observación del array.
- α es el desfase entre elementos del array.

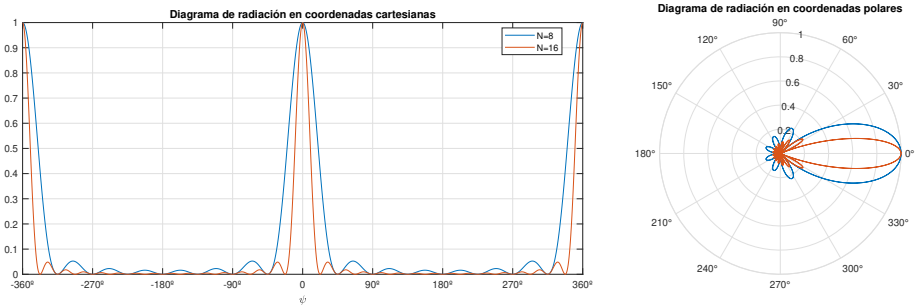


Figura 2.1: Diagrama de radiación para $d = \lambda/2$ y valores de N de 8 y 16.

Típicamente se muestra el diagrama de radiación como el módulo de $F(\psi)$ al cuadrado, como se representa en la Figura 2.1. En este ejemplo se comparan los diagramas de radiación de dos array de 8 y 16 sensores para mostrar cómo se modifica el patrón al aumentar

el número de micrófonos. Nótese que a mayor número de sensores disminuye el ancho del lóbulo principal, debido a que la longitud total del array (apertura espacial) aumenta, lo que nos permite discriminar entre blancos más cercanos, o lo que es lo mismo, aumentar la resolución espacial. Por ese mismo motivo, también es posible mejorar esta resolución aumentando la distancia d entre micrófonos.

Esquemas de implementación

Un conformador implementa el esquema de procesado mostrado en la Figura 2.2 y puede expresarse formalmente como

$$y = \sum_{i=1}^{N} w_i^* \cdot x_i \tag{2.2}$$

donde w_i son los pesos asignados al conformador y x_i representa las señales.

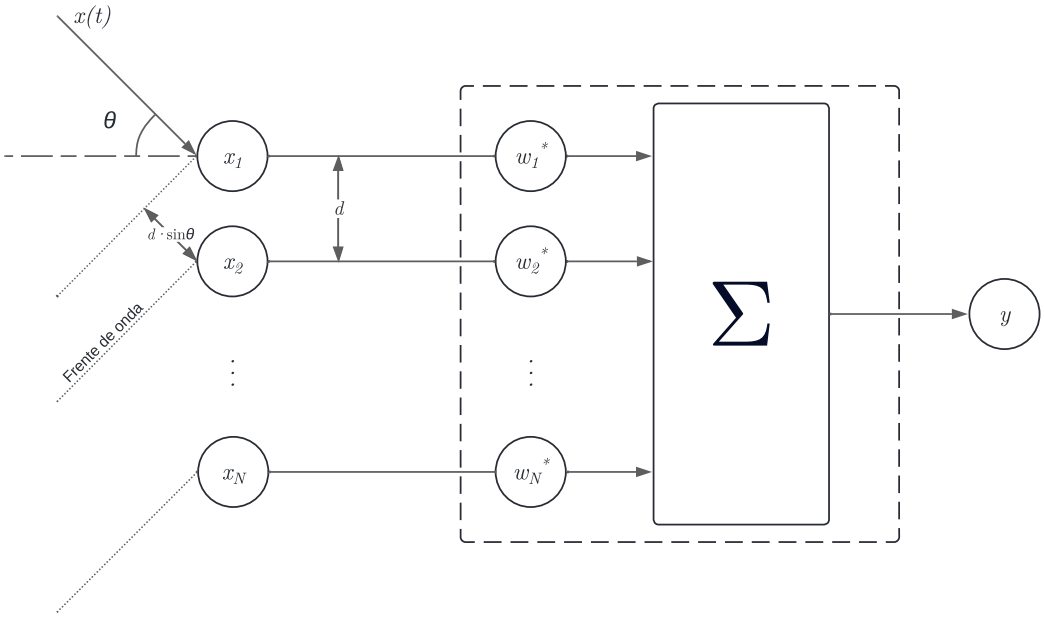


Figura 2.2: Esquema básico de un conformador

Vectorialmente se representa como

$$\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; ... \;\; x_N]^T \tag{2.3}$$

$$\mathbf{w} = [w_1 \;\; w_2 \;\; w_3 \;\; ... \;\; w_N]^T \tag{2.4}$$

$$\mathbf{y} = \mathbf{w}^H \cdot \mathbf{x} \tag{2.5}$$

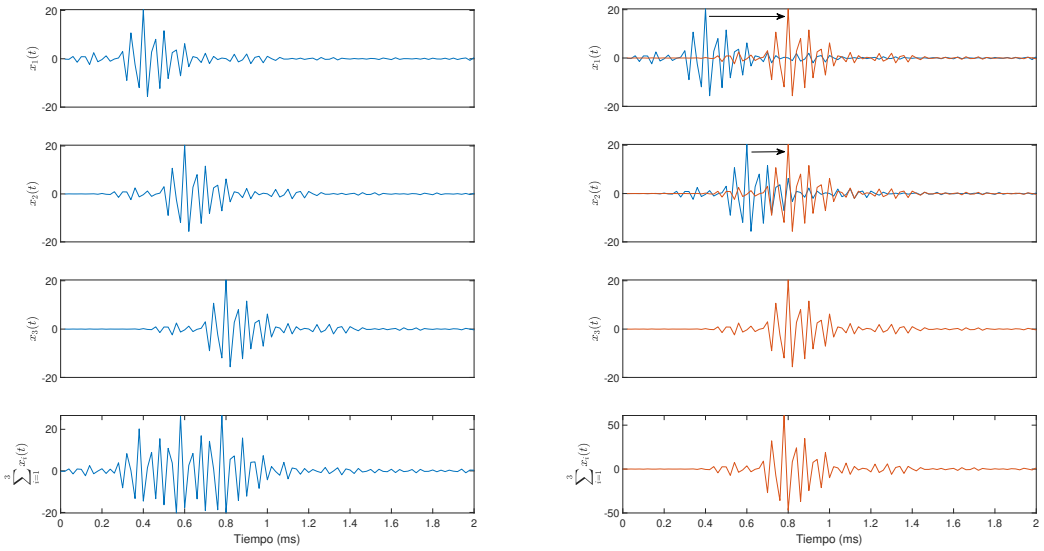
donde $\mathbf{x}$ es el vector de señales, $\mathbf{w}$ el vector de pesos, y H representa la transpuesta conjugada.

Se podría expresar como ecuación matricial de la siguiente manera

$$\begin{pmatrix} y(T_s) & y(2T_s) & \dots & y(NT_s) \end{pmatrix} = \begin{pmatrix} w_1 & w_2 & \dots & w_N \end{pmatrix} \begin{pmatrix} x_1(T_s) & x_1(2T_s) & \cdots & x_1(NT_s) \\ x_2(T_s) & x_2(2T_s) & \cdots & x_2(NT_s) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(T_s) & x_M(2T_s) & \cdots & x_M(NT_s) \end{pmatrix}$$

Existen diversas técnicas de implementación de algoritmos de *beamforming*, pero nos centraremos en una denominada *Delay and Sum*, atrasar y sumar en español.

En el caso de un conformador *Delay and Sum* asumiendo que la onda llega primero al micrófono 1, un instante τ después al micrófono 2, y así para el resto de micrófonos, si sumamos todas las señales recibidas, al no estar en fase se cancelarán parcial o totalmente entre sí como se ve en la Figura 2.3(a). Sin embargo, sabiendo que la distancia entre los micrófonos es d y que el ángulo de llegada es θ podemos calcular qué retardo hay que aplicar a cada micrófono para que todas las señales capturadas estén en fase, como se muestra en la Figura 2.3(b).



(a) Suma de las señales sin desplazar.

(b) Suma de las señales atrasadas.

Figura 2.3: Conformación basada en *Delay and Sum*.

Nótese que en 2.3(b) la amplitud de la señal resultante es aproximadamente tres veces mayor que cada señal por separado, puesto que no se ha aplicado una normalización tras la suma de las señales. Esto aporta una especie de "filtrado" adicional basado en el promedio, en el que se atenúa todas aquellas muestras que no sean comunes a todas las señales.

Esta técnica de procesado responde a la siguiente ecuación

$$b(t) = \frac{1}{M} \sum_{m=1}^{M} w_m \, x_m(t - \tau_m) \tag{2.6}$$

2.1.2. Aliasing espacial

Como ya se mencionó previamente, una de las formas de conseguir una mejor resolución espacial es aumentar la distancia entre sensores, pero podríamos preguntarnos qué ocurre si aumentamos mucho esta distancia. Si bien es cierto que el ancho del lóbulo principal disminuye, también lo hace con él el resto del diagrama de radiación, y como ya observamos en la Figura 2.1, este es 2π-periódico, vemos como en $\pm 360^{\text{o}}$ aparecen de nuevo lóbulos principales denominados *grating lobes* o lóbulos fantasma, con lo que si estos se trasladan a ángulos menores tendríamos un problema a la hora de distinguir un único ángulo de llegada.

En general, para no cometer este *aliasing* espacial se debe cumplir que la distancia entre sensores no supere

$$d_{\text{máx.}} \leq \lambda/2 \tag{2.7}$$

Otra manera de evitar la aparición de estos *grating lobes* es equiespaciar los sensores.

2.2. Acústica de insectos xilófagos

Podemos considerar la combinación de la larva y de la madera que la alberga como una fuente acústica conjunta. Por tanto, el sonido producido dependerá de ambos, por lo que es importante conocer la relación entre el comportamiento y la morfología de la larva y el posible impacto de las distintas propiedades de la madera en el sonido resultante. [2]

Transmisión de ondas acústicas en la madera

El sonido se define como la transmisión de una vibración a través de las partículas de un material [2].

La composición de un material influye en la propagación de la onda acústica en dos aspectos cruciales: la atenuación y la velocidad. La atenuación se refiere a la tasa a la que se pierde energía a medida que la onda acústica atraviesa el material, mientras que la velocidad es la rapidez con la que una onda se desplaza a través del material. En el caso de la madera, un mayor contenido en humedad hace que su densidad aumente, lo que lleva a una disminución de la velocidad del sonido [17]. Esto ocurre porque una partícula de menor masa puede moverse más que una partícula de mayor masa cuando se aplica la misma cantidad de fuerza. La densidad de la madera varía significativamente entre especies, pero también depende de otros factores. En condiciones secas, la densidad del roble puede ser entre 1 y 3 veces mayor que la del pino canadiense y hasta 5 veces mayor que la de la madera de balsa. [2]

Incluso teniendo en cuenta las diferencias entre las distintas especies, las propiedades acústicas pueden variar. La madera cortada tiende a variar su humedad con respecto a los árboles en pie, lo cual afecta tanto a la atenuación como a la velocidad del sonido en su interior. Otra de las propiedades a considerar es el grado de deterioro, que también hace variar la densidad de la madera. [2]

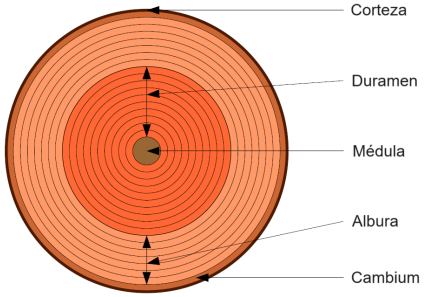


Figura 2.4: Sección transversal de un árbol. [2]

En la Figura 2.4 se muestra un corte transversal del tronco de un árbol con sus distintas regiones diferenciadas. Viendo este diagrama, puede observarse que el árbol no es un sólido continuo con una estructura homogénea, si no que presenta una compleja sucesión de capas cada una de ellas con distintas elasticidad y densidad. Aparte de las diferencias que pueda haber en la sección transversal, las vetas de la madera producen una variación en la densidad, lo que provoca un cambio de la velocidad del sonido relacionado con el ángulo formado entre la onda y la veta. [2]

Aparte de los sonidos transmitidos por la larva, en determinadas ocasiones la madera puede producir sonidos. Entre ellos, destacan las emisiones acústicas por estrés y fractura, causadas por la presión ejercida sobre las fibras de madera que puede romperlas produciendo vibraciones que viajan a través de la madera. Otros sonidos pueden provocarse en árboles en pie, dada su condición de ser un material vivo, por ejemplo el sonido provocado por cavitación, que es la formación de gases dentro de un líquido (xilema). [2]

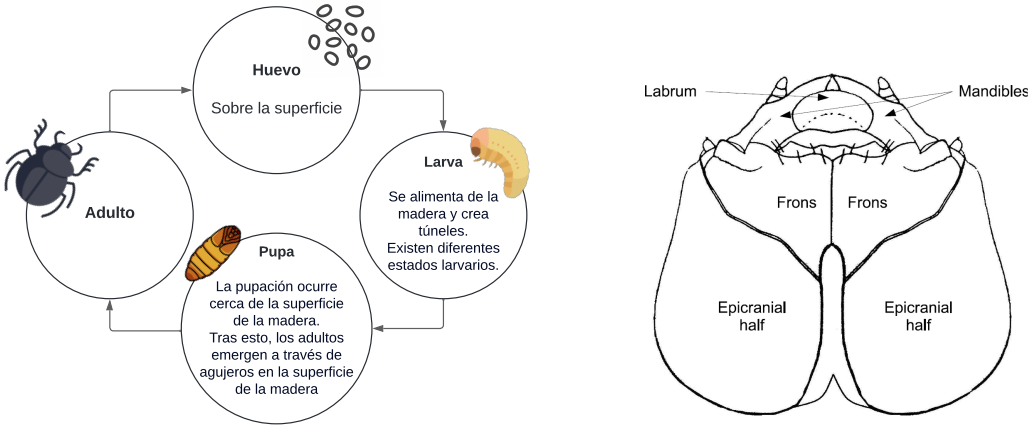
Acústica de insectos

Los insectos producen una amplia variedad de sonidos con distintos objetivos, y en algunos casos, sin objetivo alguno. Estos se pueden dividir en dos categorías distintas: sonidos accidentales, producidos por acciones realizadas por el insecto, como la alimentación; y no accidentales, aquellos provocados intencionalmente por el insecto, por ejemplo para la comunicación. [2]

Esta investigación se centra en los sonidos accidentales, concretamente en aquellos producidos por la larva al alimentarse, por lo que los sonidos dependerán directamente de la morfología y el comportamiento de la larva. También hay que tener en cuenta los ciclos de vida del insecto y su comportamiento en cada uno de ellos, puesto que su actividad no será la misma. Aunque existen diferencias entre especies, y en ocasiones también entre individuos de la misma especie, en función de factores externos como por ejemplo la temperatura, en la Figura 2.5(a) se muestra el ciclo vital de un insecto xilófago.

Puesto que no hay evidencia de que los insectos en su fase de huevo y pupa produzcan algún sonido de interés [2], únicamente se estudiarán los sonidos producidos al alimentarse durante el estado larvario. Para entender la naturaleza de estos sonidos es útil comprender ciertas propiedades y características de la larva.

Cuando el insecto está en fase larval, su principal actividad es alimentarse. La larva se nutre de la madera creando un túnel en su interior durante este proceso. Para ello, emplea sus mandíbulas muy esclerotizadas, ilustradas en la Figura 2.5(b). Al alimentarse la larva, usa sus mandíbulas para agarrar y romper las fibras de la madera, cuya rotura genera una vibración que se propaga a través de la madera permitiendo su detección vía acústica.



(a) Ciclo vital de un insecto xilófago. (b) Diagrama de la cabeza de un cerambícido.

Figura 2.5: Ciclo vital y morfología mandibular de un insecto xilófago. [2]

En la Figura 2.6 se muestra la forma de onda típica de una mordedura producida por un ejemplar de *Hylotrupes bajulus*, muestreada a 50kHz.

No se conoce mucho sobre la variación en el comportamiento de la alimentación de las larvas y los sonidos asociados con ellos, sin embargo, es probable que este comportamiento esté condicionado por diversos factores internos y externos.

Se ha observado que la mayoría de las larvas pasan un periodo de varios meses previos a la pupación sin alimentarse [18], lo cual deja entrever un problema asociado a la detección vía acústica. En este caso, únicamente serían útiles métodos de detección activa como los basados en rayos X o reflexión de ultrasonidos.

En cuanto a las diferencias entre especies, en [19] se demostró la viabilidad de clasificar dos especies, *Hylotrupes bajulus* y *Prionus coriarius*, en base a su patrón de ocurrencia con una tasa de éxito del 96 %, lo cual pone de manifiesto la gran diferencia entre especies en dicho patrón.

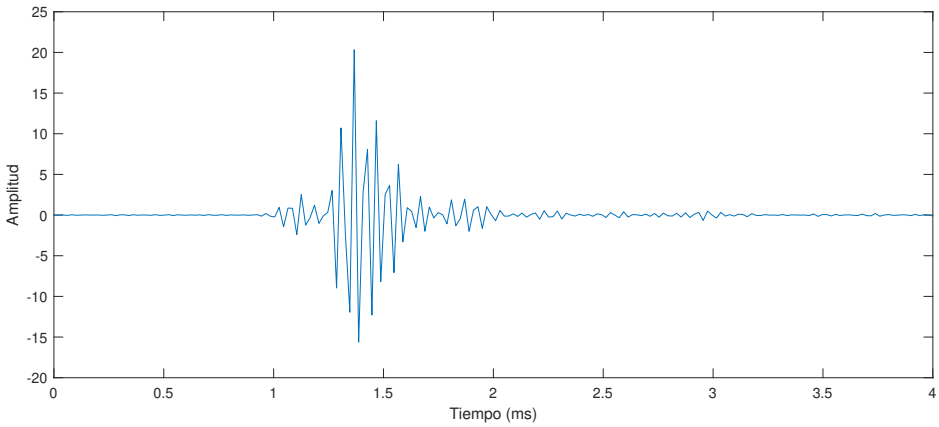


Figura 2.6: Una mordida de *Hylotrupes bajulus L.*.

2.3. Deep learning y Redes Neuronales

En los últimos años ha aumentado el interés por conceptos como Inteligencia Artificial (IA), Machine Learning (ML, también conocido como aprendizaje máquina o aprendizaje automático) o Deep Learning (DL, también denominado aprendizaje profundo). Sin embargo, es importante distinguir estos términos y conocer su relación antes de entrar en más detalle.

La IA nació en la década de 1950, cuando en el ámbito de las Ciencias de la computación empezaron a preguntarse si sería posible hacer a los ordenadores "pensar" [20]. Existen diversas definiciones de IA, pero una de ellas es "*El esfuerzo de automatizar tareas intelectuales típicamente llevadas a cabo por humanos*" [20]. Como tal, la IA es un campo de estudio muy amplio que engloba ML y DL, pero también abarca otros enfoques que no tienen que ver con el aprendizaje. [20]

Por su parte, el aprendizaje automático nace de la idea de que un ordenador puede aprender a realizar una tarea sin tener que crear un programa con unas estrictas reglas definidas a mano, lo cual abre un nuevo paradigma en la programación. Un sistema de ML es entrenado, lo que equivale a programarlo en un paradigma clásico, esto es presentarle ejemplos relevantes de una tarea a realizar, y que él encuentre patrones estadísticos que finalmente le permitan definir unas reglas para automatizar la tarea. [20]

Sin embargo, para este proyecto el interés está en el campo del aprendizaje profundo. Como se muestra en la Figura 2.7, el DL es un subcampo de ML, el cual a su vez es parte del campo de la IA. La diferencia de DL respecto a ML es que supone una nueva forma de aprender representaciones a partir de datos que hace hincapié en el aprendizaje de capas sucesivas de representaciones cada vez más significativas. [20]

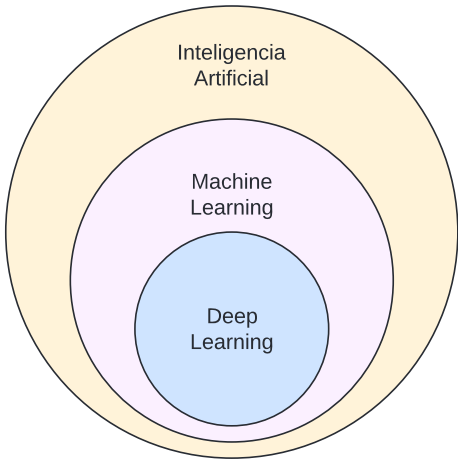


Figura 2.7: Inteligencia artificial, machine learning y deep learning.

2.3.1. Redes neuronales artificiales

Este concepto previamente mencionado de capas está directamente relacionado con las redes neuronales. Las redes neuronales artificiales (ANNs, del inglés *Artificial Neural Networks*) fueron inspiradas por sus análogas biológicas [21] y son capaces de descubrir relaciones interesantes en un conjunto de datos [22].

En 1943 McCulloch y Pitts elaboraron un modelo simplista de una neurona mostrado en la Figura 2.8. Este modelo define que cada neurona está formada por un interruptor que recibe una señal de entrada de otras neuronas, y en función de si la suma total de las entradas supera o no un umbral, se activa o no [23]. La salida de estas neuronas artificiales se puede formular matemáticamente de la siguiente manera

$$\hat{y} = g\left(\sum_{i=1}^{N} x_i w_i + b\right) \tag{2.8}$$

donde

- $\hat{y}$ es la salida de la neurona.
- g es la función de activación.
- N es el número de entradas a la neurona.
- x_i es cada entrada a la neurona.
- w_i es el peso asociado a cada x_i.
- b es un término constante denominado sesgo o *bias*.

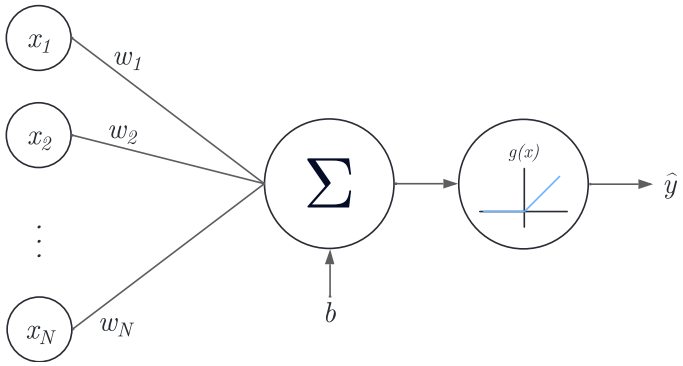


Figura 2.8: Esquema de una neurona artificial.

Estructura de una Red Neuronal

Las redes neuronales profundas de propagación directa, también llamadas redes neuronales de propagación directa (*feed-forward*) o perceptrones multicapa (MLP, por sus siglas en inglés), son los modelos esenciales del aprendizaje profundo [24]. El objetivo de una red de propagación directa es aproximar alguna función que, por ejemplo, mapee una entrada x en una categoría y para el caso de un problema de clasificación.

Estos modelos reciben el nombre de propagación directa porque los datos fluyen desde la entrada x a través de las capas intermedias hasta la salida y, sin conexiones hacia atrás [24]. Las redes neuronales de propagación directa reciben el nombre de redes porque típicamente se representan como una composición de diferentes funciones. Por ejemplo, supongamos que tenemos tres funciones $f^{(1)}$, $f^{(2)}$ y $f^{(3)}$ conectadas en cadena de manera que $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. En este caso $f^{(1)}$ sería la primera capa de la red, $f^{(2)}$ la

segunda y así sucesivamente. La longitud total de la cadena determina la profundidad del modelo, de ahí el nombre de aprendizaje profundo.

Una red neuronal típicamente está formada por una capa de entrada, en el caso de una señal temporal con una neurona por muestra, a continuación una serie de "capas ocultas", variables en función del problema a resolver, y por último una o más neuronas a en la capa de salida. En el caso de un problema de clasificación las neuronas de la última capa serán tantas como clases haya, con una única neurona si se trata de un problema de clasificación binaria y varias si realiza una tarea de clasificación multiclase.

Las funciones de activación son un aspecto crítico para el correcto funcionamiento de una red neuronal. Existen diversas formas, pero a continuación se presentan dos de las más habituales y usadas en este proyecto, cuyas gráficas se representan en la Figura 2.9.

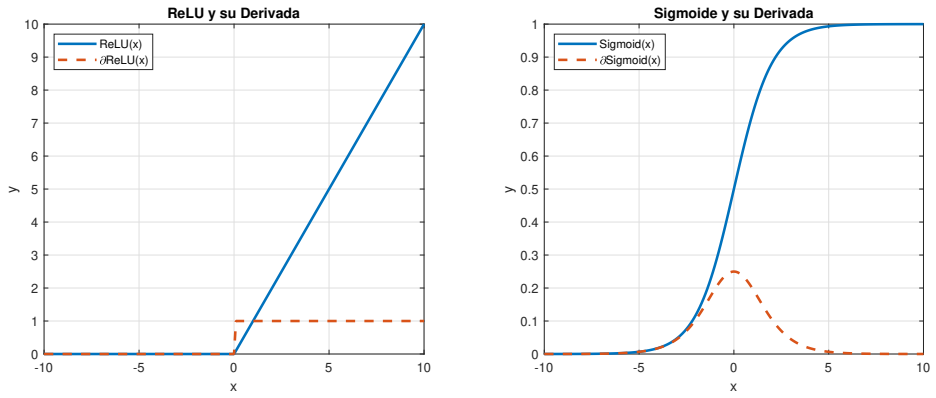


Figura 2.9: Funciones de activación y sus derivadas.

1. ReLU o rectificador: es una función de activación no lineal ampliamente usada. Hace que la neurona se desactive solo cuando la salida de la transformación lineal sea cero [25]. Se puede definir matemáticamente como

$$f(x) = \max(0, x) \tag{2.9}$$

   La función ReLU es más eficiente que otras porque no todas las neuronas se activan al mismo tiempo, sino que solo un determinado número de ellas lo hacen. [25]

   Sin embargo, en algunos casos su derivada es cero, lo que hace que sus pesos y sesgos no se actualicen durante la retropropagación (véase 2.3.4). Para reducir este problema se han creado otras funciones similares ligeramente modificadas, cuya derivada no es siempre cero para valores negativos, son ejemplo las funciones *Leaky ReLU* o *Exponential Linear Unit* (ELU).

2. Sigmoide. Se trata de una función de activación no lineal muy empleada [25] que se puede definir matemáticamente como

$$f(x) = \frac{1}{e^{-x}} \tag{2.10}$$

   Se suele emplear en la neurona de salida de un clasificador binario puesto que tiende a asignar un valor de 1 para entradas positivos, y 0 en aquellos caso que la entrada es negativa.

Existen otro tipo de capas con el objetivo de mejorar la generalización del modelo y reducir el sobreajuste. Es el caso de las capas de *dropout* que permiten desactivar una serie de neuronas durante el una iteración de forma aleatoria con el objetivo de crear una red menos sensible a pesos específicos, mejorando la eficiencia computacional puesto que la red se vuelve más pequeña. Sin embargo, el uso de estas técnicas de regularización requieren de más pasos para converger.

Además, en el entrenamiento de una red neuronal no solo influyen las capas que componen su arquitectura, sino que hay una serie de *hiperparámetros* que afectan al resultado. Entre ellos destacan los siguientes:

1. Época o *epoch*, que representa una iteración de entrenamiento en la que el modelo ve todo el conjunto de datos de entrenamiento. Si el número de épocas es muy pequeño el modelo puede no aprender bien las características de los datos, llevando a un problema de subajuste, mientras que un número muy elevado de épocas puede hacer que el modelo memorice estas características ocasionando un problema de sobreajuste.

2. Lote o *batch*. Un lote es un conjunto de ejemplos que se evalúan de manera conjunta durante el entrenamiento. El procesamiento por lotes mejora la eficiencia computacional y permite actualizar los pesos de la red más suavemente. [26]

3. Tasa de aprendizaje. Este valor mide cómo de rápido se adapta el modelo a los nuevos datos. Un valor alto de tasa de aprendizaje hará que tu modelo cambie rápidamente ajustándose muy bien al nuevo dato, pero tenderá a olvidar lo aprendido con datos anteriores. [21]

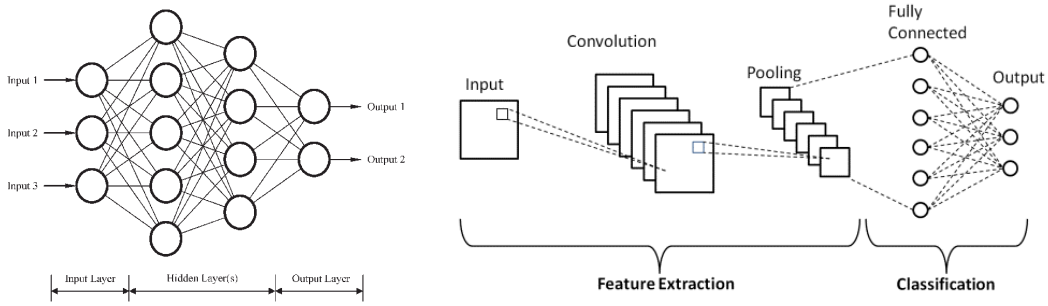
2.3.2. Arquitecturas de redes neuronales

Las redes neuronales se basan en neuronas artificiales, que unidas entre sí forman una red. Hoy en día el número de capas puede oscilar desde unas pocas a miles de ellas, y generalmente se emplea el termino de red neuronal densa (DNN, del inglés *Deep Neural Network*) a las redes neuronales usadas en DL [27]. Estas redes suelen contener varias capas ocultas lo que le otorga la capacidad de extraer características de distinto nivel en cada una de las sucesivas capas [28].

Perceptrón Multicapa

Un perceptrón multicapa (MLP, del inglés *Multilayer Perceptron*) es un tipo de red neuronal compuesto por una capa de entrada, una o más capas ocultas y una capa final de salida. Las capas cercanas a la capa de entrada se suelen denominar capas bajas, mientras que aquellas cercanas a la capa de salida reciben el nombre de capas altas [21]. Todas las neuronas, excepto las de la capa de salida, incluyen un sesgo y están completamente conectadas con la siguiente capa, lo que se conoce como *fully connected* (FC).

Los MLP son comúnmente utilizados en tareas de clasificación tanto binaria como multiclase. Presentan una limitación y es que no son capaces de clasificar patrones que no sean separables linealmente [29] a no ser que el número de capas ocultas sea muy elevado. En la Figura 2.10(a) se muestra un ejemplo de MLP.



(a) Perceptrón multicapa. [30]

(b) Red Neuronal Convolucional.

Figura 2.10: Arquitecturas de redes neuronales.

Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales (CNN, del inglés *Convolutional Neural Network*) son un tipo de DNN que se utilizan principalmente en tareas como el reconocimiento de imágenes, la detección de objetos y la segmentación de imágenes, aprovechando su capacidad para aprender y reconocer patrones espaciales.

Una CNN normalmente está compuesta por tres tipos de capas diferentes:

1. Convolucionales. Representan una parte crucial de esta arquitectura y es un conjunto de filtros, o kernels, que permiten extraer características de los datos. Los valores de estos filtros se calculan durante el entrenamiento de la red [31]. Estos filtros se aplican a cada píxel transformando la imagen original en tantas imágenes filtradas como filtros haya, construyendo un mapa de características.

2. Pooling. Esta capa, también conocida como capa de reducción de muestreo, sirve para disminuir la dimensión de los mapas de características, conservando la información más relevante. Hay varios tipos, entre los que destaca la capa *max*, que se queda con el valor de píxel más alto de un entorno, y la capa *avg*, que hace la media de los valores de los píxeles de ese entorno. [31]

3. *Fully Connected*. Una vez extraídas las características y reducida su dimensión, se conecta la salida a una red neuronal completamente conectada como se muestra en la Figura 2.10(b), encargada de realizar la tarea de clasificación.

A pesar de su gran utilidad para la visión por ordenador y tareas como la de clasificación de imágenes, su complejidad hace que sea necesario entrenar muchos más parámetros respecto a una arquitectura más simple como un MLP, lo que hace que el entrenamiento sea más lento y requiera una mayor capacidad de computo.

2.3.3. Métodos de Optimización

Casi todos los algoritmos de ML pueden formularse como un problema de optimización en el que encontrar el extremo de una función objetivo. Para alcanzarlo, en numerosas ocasiones se recurre a métodos de optimización numéricos o analíticos [32], algunos de los cuales se explican a continuación.

Descenso del gradiente

El descenso del gradiente es una forma de minimizar una función objetivo $J(\theta)$ parametrizada por los parámetros de un modelo $\theta \in \mathbb{R}^d$ actualizando los parámetros en dirección opuesta al gradiente de la función objetivo $\nabla_\theta J(\theta)$ con respecto a los parámetros. La tasa de aprendizaje η determina el tamaño de los pasos que tomamos para alcanzar un mínimo (local). [33]

Existen variantes de este algoritmo entre las cuales destacan las siguientes:

1. El gradiente descendiente batch calcula el gradiente de la función de coste, que es la diferencia entre el valor estimado y el valor real, con referencia a los parámetros θ para todo el conjunto de datos según la expresión:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.11}$$

   Al tener que calcular los gradientes para todo el conjunto de datos para hacer solamente una actualización, este método es muy lento. [33]

2. El gradiente descendiente estocástico hace una actualización de los parámetros para cada ejemplo de entrenamiento $x^{(i)}$ y etiqueta $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.12}$$

   Suele ser más rápido que el gradiente descendiente batch y permite encontrar nuevos y mejores mínimos locales. [33]

3. El gradiente descendiente mini-batch mezcla lo mejor de los anteriores actualizando los parámetros para cada mini-batch de n ejemplos de entrenamiento:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i;i+n)}; y^{(i;i+n)}) \tag{2.13}$$

   De esta forma se reduce la varianza en las actualizaciones lo que puede llevar a una convergencia más estable y puede hacer uso de librerías que calculan en gradiente respecto a un mini-batch de manera muy eficiente. [33]

Optimizadores avanzados

A pesar de que el descenso de gradiente es un optimizador muy utilizado, y para una gran mayoría de casos sea suficiente, presenta algunos problemas. Elegir una tasa de aprendizaje puede ser difícil, puede ser deseable no actualizar todos los parámetros de la misma manera si los datos son dispersos o puede haber problemas con los mínimos locales o puntos de silla que no permitan alcanzar un mínimo global óptimo. Es por eso que se han desarrollado otros algoritmos más avanzados como Adam, RMSprop o Adagrad. [33]

De entre ellos, ADAM (*Adaptive Moment Estimation*) es el que se ha empleado en este trabajo. Adam es un algoritmo de optimización diseñado para entrenar redes neuronales profundas de manera eficiente y efectiva. Combina los beneficios del método de gradientes estocásticos con técnicas de optimización adaptativas. En esencia, Adam mantiene dos promedios exponenciales de los gradientes y los cuadrados de los gradientes, lo que le permite ajustar automáticamente las tasas de aprendizaje para cada parámetro. Además,

incluye términos de corrección de sesgo para compensar el sesgo inicial de los estimadores de los momentos. Estos mecanismos ayudan a mantener una tasa de aprendizaje adecuada para cada parámetro durante el entrenamiento, lo que acelera la convergencia y mejora el rendimiento del modelo en una amplia gama de problemas de aprendizaje automático. [34]

El algoritmo seguido por este optimizador es el siguiente:

1. Primero, se inicializan los parámetros a 0.

$$m_0 = 0, \quad v_0 = 0, \quad t = 0 \tag{2.14}$$

2. Después para cada iteración t:

$$t = t + 1 \tag{2.15}$$

$$g_t = \nabla f(\theta_{t-1}) \tag{2.16}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.17}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.18}$$

   donde g_t es el gradiente en el tiempo t, y β_1 y β_2 son los coeficientes de decaimiento para los momentos de primer y segundo orden, respectivamente.

3. Después, para corregir los sesgos introducidos por las inicializaciones de m y v:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.19}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.20}$$

4. Finalmente, se actualizan los parámetros θ:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.21}$$

   donde α es la tasa de aprendizaje, y ϵ un término pequeño para evitar la división por cero.

2.3.4. Retropropagación

El algoritmo de aprendizaje por retropropagación descrito por primera vez en [35] funciona para redes de alimentación hacia adelante (*feed-forward*) con salidas continuas. El entrenamiento comienza estableciendo todos los pesos de la red en valores pequeños y aleatorios. Ahora, para cada ejemplo de entrada, la red genera una salida, que inicialmente es aleatoria. Se mide la diferencia cuadrática entre esta salida y la deseada, que representa la clase o valor correcto. La suma de todas estas diferencias cuadráticas a lo largo de todos los ejemplos de entrenamiento se denomina el error total de la red. Si este valor fuese cero, la red sería perfecta, y cuanto menor sea el error, mejor será el resultado de la red. [23]

Eligiendo los pesos que minimicen el error total, se puede obtener la red neuronal que mejor resuelve el problema en cuestión, y para eso se hace uso del siguiente algoritmo. [21]

1. El algoritmo maneja un mini-lote a la vez y recorre el conjunto de entrenamiento completo múltiples veces.

2. Cada mini-lote se pasa a la capa de entrada de la red, que simplemente lo envía a la primera capa oculta. El algoritmo luego calcula la salida de todas las neuronas en esta capa (para cada instancia en el mini-lote). El resultado se pasa a la siguiente capa, se calcula su salida y se pasa a la siguiente capa, y así sucesivamente hasta obtener la salida de la última capa, la capa de salida. Este es el paso hacia adelante.

3. A continuación, el algoritmo mide el error de salida de la red.

4. Luego calcula cuánto contribuyó cada conexión de salida al error. Esto se hace analíticamente aplicando simplemente la regla de la cadena, lo que hace que este paso sea rápido y preciso.

5. El algoritmo luego mide cuántas de estas contribuciones de error provinieron de cada conexión en la capa inferior, nuevamente usando la regla de la cadena, y así sucesivamente hasta que el algoritmo llega a la capa de entrada.

6. Finalmente, el algoritmo realiza un paso de descenso del gradiente para ajustar todos los pesos de conexión en la red, utilizando los gradientes de error que acaba de calcular.

En la Figura 2.11 se presenta un diagrama explicativo con el objetivo de clarificar el funcionamiento de este algoritmo.

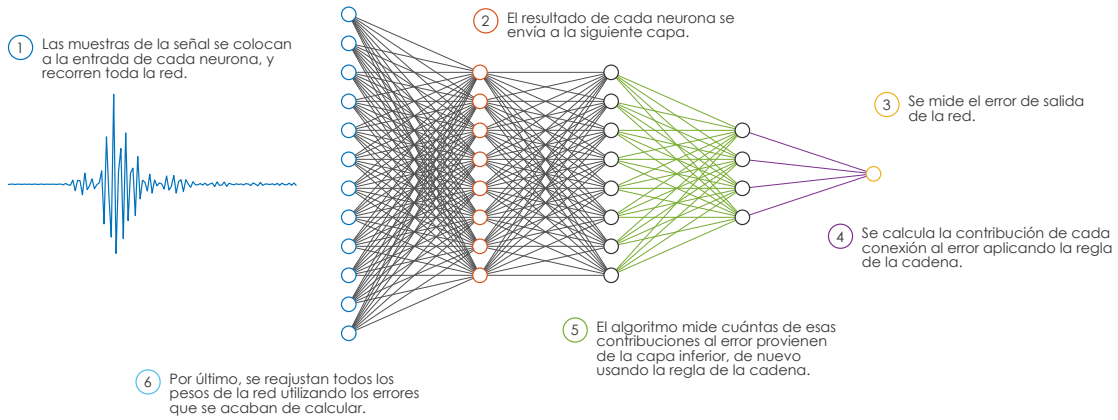


Figura 2.11: Algoritmo de retropropagación

2.4. Métricas de rendimiento en algoritmos de *deep learning*

Uno de los componentes críticos de DL es la elección de la métrica de rendimiento usada para entrenar y evaluar los modelos. Las distintas métricas de rendimiento evalúan la habilidad del modelo para hacer predicciones correctas sobre datos nunca antes vistos.

La elección de una métrica es crucial para alcanzar un buen rendimiento, sin embargo, debido a la gran variedad de opciones disponibles, puede ser complicado elegir el método

más apropiado para nuestra tarea específica. A continuación se definen las métricas más usadas en el ámbito de DL. [36]

2.4.1. Matriz de confusión

Una herramienta ampliamente usada a la hora de evaluar el rendimiento de un algoritmo de clasificación es la matriz de confusión. En ella se muestra el número de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) resultantes de evaluar el algoritmo. La matriz de confusión para un problema de clasificación binaria se presenta en la tabla 2.1.

	Positivo predicho	**Negativo predicho**
Positivo real	Verdaderos positivos (TP)	Falsos negativos (FN)
Negativo real	Falsos positivos (FP)	Verdadero negativo (TN)

Tabla 2.1: Matriz de confusión

Donde, para nuestro caso particular:

- TP es el número de nudos identificados correctamente como nudos.
- TN es el número de larvas correctamente clasificadas como larvas.
- FN es el número de nudos que han sido mal identificados como larvas.
- FP es el número de larvas que han sido clasificados erróneamente como nudos.

2.4.2. Exactitud

La exactitud o *accuracy* es una de las métricas más usadas en el ámbito de la clasificación. Es el ratio de las muestras clasificadas correctamente entre el número total de muestras. Se puede expresar como

$$\text{Exactitud} = \frac{\text{Número de muestras clasificadas correctamente}}{\text{Número de muestras total}} \tag{2.22}$$

También se puede expresar en términos de los valores de la matriz de confusión como

$$\text{Exactitud} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.23}$$

Se trata de una métrica simple e intuitiva, pero que puede llevar a error cuando la distribución de clases no está balanceada, puesto que tiende a favorecer a la clase mayoritaria. Para esos casos, existen otras métricas más apropiadas como la precisión, la exhaustividad o la puntuación F1.

2.4.3. Precisión

La precisión mide la exactitud de las predicciones positivas, es decir, de las muestras clasificadas en la clase positiva cuántas pertenecen realmente a dicha clase. En nuestro problema de clasificación, los nudos que se han identificado correctamente como tal. Se define como el número de predicciones positivas verdaderas frente al número total de predicciones positivas, tal y como se expresa a continuación

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2.24}$$

La precisión es útil cuando el coste de un falso positivo es elevado. Un valor alto de esta métrica nos indica que el modelo no está generando muchos falsos positivos, por lo que las predicciones son fiables. Sin embargo, es importante considerar métricas adicionales, puesto que un modelo que no genere muchas predicciones positivas podría conseguir un alto valor de precisión, aunque mostraría una exhaustividad baja. [36]

2.4.4. Exhaustividad

La exhaustividad o *recall*, también conocida como sensibilidad o Tasa Positiva Verdadera (TPR, del inglés *True Positive Rate*), es una métrica que mide la proporción de muestras de verdaderos positivos del total de muestras positivas, en partícular, la proporción de nudos clasificados correctamente del total de muestras clasificadas como nudo. Matemáticamente sigue la expresión:

$$\text{Exhaustividad} = \frac{TP}{TP + FN} \tag{2.25}$$

Mide cómo de bien identifica el modelo todas las muestras positivas del conjunto de datos. Un valor alto de esta métrica indica que el modelo tiene pocos falsos negativos, lo que significa que puede identificar correctamente la mayoría de las muestras positivas. Sin embargo, una exhaustividad alta no implica necesariamente que el modelo tenga una precisión alta. Hay una relación de compromiso entre precisión y exhaustividad puesto que están inversamente relacionadas, cuando una aumenta, la otra disminuye. [36]

2.4.5. Tasa de falsos positivos

La Tasa de Falsos Positivos (FPR, del inglés *False Positive Rate*) se usa para evaluar la proporción de falsos positivos, es decir, aquellas muestras de larva que se han clasificado incorrectamente como nudo respecto al número total de larvas. También se denomina tasa de error tipo I, y es complementario a la especificidad. [36]

Formalmente, se calcula como:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.26}$$

2.4.6. Puntuación F1

Existe una métrica que combina la precisión y la exhaustividad proporcionando un único valor representativo del rendimiento global de la clasificación del modelo. Se trata de la puntuación F1 y se define como la media armónica de la precisión y la exhaustividad, la cual se calcula como sigue

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}} \tag{2.27}$$

La puntuación F1 considera tanto la habilidad del modelo para identificar correctamente los ejemplos positivos (precisión) como la de identificar todos los ejemplos positivos del conjunto de datos (exhaustividad). Es especialmente útil cuando la distribución entre clases no es balanceada, o cuando queremos dar el mismo peso a la precisión y a la exhaustividad. [36]

2.4.7. Especificidad

La especificidad, también conocida como Tasa Negativa Verdadera, es una métrica que mide la proporción de verdaderos negativos correctamente identificados por el modelo. Se define como el número de verdaderos negativos entre el número total de negativos reales. Su expresión matemática es

$$\text{Especificidad} = \frac{TN}{TN + FP} \tag{2.28}$$

Es especialmente importante en ámbitos donde es crítico minimizar el número de falsos negativos para evitar tomar acciones innecesarias que puedan suponer un problema [36]. Sin embargo, en el caso de la detección de larvas, no es crucial que el algoritmo clasifique algún sonido que no corresponda a una larva como tal, puesto que se tomarán múltiples muestras, lo que hará que estos falsos negativos sean residuales y no afecten significativamente los resultados finales.

2.4.8. Curva de Características Operativas del Receptor (ROC)

La curva ROC es una herramienta estadística usada para evaluar la capacidad discriminatoria de una prueba, en este caso, un algoritmo de clasificación. En estas curvas se representa la sensibilidad en función de los falsos positivos (complementario de la especificidad) para distintos puntos de corte. [37]

Una curva ROC que sigue una línea diagonal $y = x$ produce igual número de falsos positivos que de verdaderos positivos. Por tanto, es deseable que la curva ROC de nuestro algoritmo se aproxime tanto como sea posible a la esquina superior izquierda, por encima de la "línea de referencia" $y = x$, como se muestra en la Figura 2.12. [38]

Para hacer el cálculo de la curva ROC el algoritmo a seguir es el siguiente: [36]

1. Se parte de un clasificador que proporcione una salida binaria y estime la probabilidad de la clase positiva. Esta probabilidad se conoce como "puntuación".

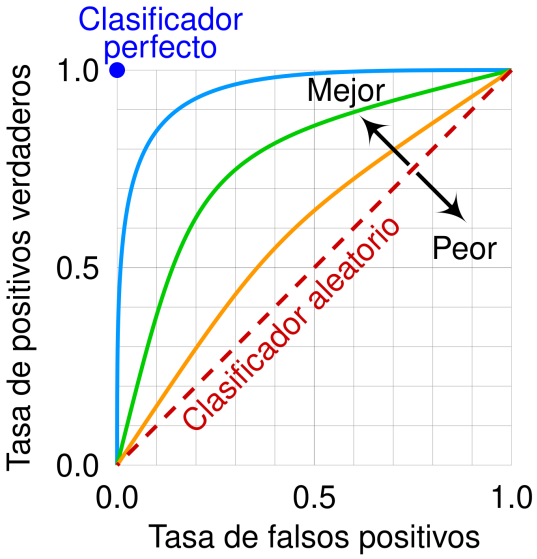


Figura 2.12: Ejemplo de curva ROC.

2. Para cada umbral posible (de 0 a 1) de estas puntuaciones, se calcula la tasa de verdaderos positivos (véase 2.4.4) y la tasa de falsos positivos (véase 2.4.4).

3. Se dibuja el valor de FPR en el eje X y el de TRP en el eje Y. Así, se obtienen gráficas como las mostradas en la Figura 2.12.

Área bajo la curva (AUC)

Asociado a la curva ROC, el área bajo esta curva (AUC, del ingés *Area Under the Curve*) es una métrica comúnmente empleada para evaluar el rendimiento del modelo de clasificación binaria. Sus valores varían entre 0 y 1, siendo 1 el valor asociado a un clasificador perfecto y 0.5 el correspondiente a un clasificador cuyo rendimiento no es mejor que una suposición al azar. [36]

2.5. Funciones de pérdida en algoritmos de *deep learning*

Una función de pérdida y una métrica de rendimiento se utilizan para evaluar el desempeño de un modelo de aprendizaje profundo, pero sirven para propósitos diferentes. A diferencia de las métricas de rendimiento, una función de pérdida se utiliza durante el entrenamiento para optimizar los parámetros del modelo. Mide la diferencia entre las salidas predichas y las esperadas del modelo, y el objetivo del entrenamiento es minimizar esta diferencia. [36]

Existen diversas funciones de pérdida, pero en este trabajo se emplea la entropía cruzada binaria o pérdida logarítmica, normalmente conocida por su nombre inglés *Binary Cross-Entropy*. En una clasificación binaria, la clase verdadera generalmente se representa mediante un vector donde la clase verdadera tiene un valor de 1 y la otra clase tiene un valor de 0. La probabilidad predicha por el clasificador se representa mediante un vector de probabilidades para cada clase, donde esta probabilidad para la clase verdadera se denota

por $p(y = 1|x)$ y la probabilidad correspondiente a la otra clase se denota por $p(y = 0|x)$.

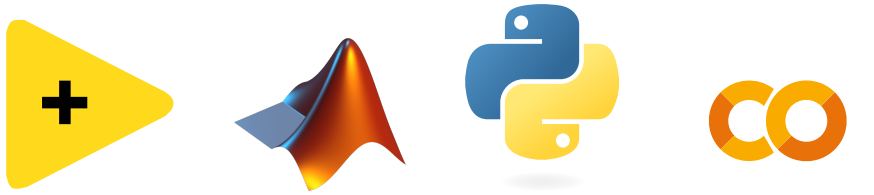
Esta función de pérdida se define como

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2.29}$$

donde y es la etiqueta de la clase verdadera (0 o 1) y p es la probabilidad predicha para la clase positiva. La función de pérdida se minimiza cuando la probabilidad predicha p es igual a la etiqueta de clase verdadera y. [36]

2.6. Herramientas de desarrollo

Para desarrollar este proyecto se ha hecho uso de las herramientas de trabajo enumeradas a continuación, cuyos logotipos se muestran en la Figura 2.13.



(a) LabVIEW™ (b) MATLAB® (c) Python (d) Google Colab

Figura 2.13: Herramientas de desarrollo.

1. El entorno de programación gráfica LabVIEW para la segmentación y el procesamiento de las señales capturadas.

2. La plataforma de programación MATLAB para el cálculo de parámetros específicos de las señales, así como para la creación de gran parte de los gráficos de esta memoria.

3. El lenguaje de programación Python para el manejo de los datos con bibliotecas como NumPy y Pandas, y para el desarrollo de las redes neuronales haciendo uso de la biblioteca TensorFlow.

4. El entorno de ejecución "Google Colab" para el entrenamiento de las redes neuronales propuestas aprovechando sus servicios de GPU.

2.6.1. LabVIEW

LabVIEW™ es el acrónimo ***Lab****oratory* ***V****irtual* ***I****nstrumen* ***E****ngineering* ***W****orkbench*. Es tanto un lenguaje como un entorno de programación gráfica lanzado en 1983 por la empresa National Instruments.

Inicialmente fue pensado para aplicaciones de control de equipos electrónicos usados en el desarrollo de sistemas de instrumentación, lo que se conoce como instrumentación virtual. Por eso los programas desarrollados en LabVIEW se guardan en ficheros VI (*Virtual Instrument)* [39]. Internamente está basado en un lenguaje de programación bien definido,

Área	Herramientas
Cálculo	Diferenciación, integración, límites, sumatorios y series de Taylor.
Álgebra lineal	Determinantes, autovalores, descomposición en valores singulares...
Solución de ecuaciones	Soluciones numéricas y simbólicas a ecuaciones algebraicas y diferenciales.
Transformadas	Fourier, Laplace, Transformada-Z y sus correspondientes transformadas inversas.

Tabla 2.2: Funciones de MATLAB en computación matemática [43]

conocido como "G", a pesar de que típicamente se usa LabVIEW para referirse tanto al entorno de desarrollo integrado (IDE) como al lenguaje de programación. [40]

La interfaz gráfica de LabVIEW cuenta con dos componentes principales: un Panel Frontal, donde estarán todos los botones, pantallas, etc. de la interfaz de usuario y una circuitería interna, definida a modo de diagrama de bloques.

2.6.2. MATLAB®

En sus inicios, MATLAB no se ideó como un lenguaje de programación, si no que era una calculadora matricial interactiva. Sin embargo, en 1984 PC-MATLAB se presentó como un sistema completo escrito en C. [41]

Actualmente, MATLAB es un sistema interactivo basado en matrices para el cálculo numérico y la visualización en ciencia e ingeniería [42] con una gran acogida en muchos centros de educación e investigación de todo el mundo. Una de sus ventajas es el hecho de poder resolver problemas numéricos complejos de una forma mucho más rápida y amigable que con lenguajes de programación como C gracias a su interfaz intuitiva y la gran variedad de *toolbox* disponibles.

MATLAB está orientado a un uso científico o en el ámbito de la ingeniería y facilita diversas tareas de computación matemática [43]. En la Tabla 2.2 se detallan algunos ejemplos de las ventajas que aporta MATLAB en determinadas áreas de la computación matemática.

Para el desarrollo de este proyecto han sido útiles algunos complementos adicionales a la versión base de MATLAB, como el "Signal Processing toolbox" o el "Wavelet toolbox", entre otros.

2.6.3. Python

Actualmente, Python es el lenguaje de programación más utilizado en el desarrollo de algoritmos de inteligencia artificial. Creado a finales de la década de 1980 por Guido Van Rossum, Python destaca por ser una herramienta potente que soporta múltiples paradigmas de programación, incluyendo la programación funcional, procedimental y orientada a objetos [44]. Su uso es extremadamente variado, pero ha ganado una enorme popularidad en el sector de la inteligencia artificial principalmente debido a la facilidad y simplicidad de su sintaxis en comparación con otros lenguajes como Lisp, Prolog, C++ o Java.

Python es especialmente atractivo por el grado de desarrollo de algunas librerías ampliamente utilizadas. Entre las más conocidas destacan NumPy para realizar operaciones algebraicas N-dimensionales, Pandas para trabajar con tablas de datos y Tensorflow, diseñado por Google para aplicar conceptos de aprendizaje automático y aprendizaje profundo de la manera más sencilla posible. Existen muchas más librerías, pero estas tres han sido las principales para el desarrollo de este proyecto.

A pesar de que existen desventajas asociadas al uso de Python, como una ejecución más lenta, un mayor uso de memoria o los problemas asociados al tipado dinámico de los datos [44], su simplicidad y apoyo en librerías con un gran soporte la convierte en una herramienta líder en el sector del desarrollo de IA.

2.6.4. Google Colab

En 2017 Google lanzó "Google Colaboratory", mejor conocido como "*Colab*". Esta plataforma permite escribir y ejecutar código Python a través del navegador y es especialmente útil para tareas de educación, análisis de datos y ML. [45]

Colab permite crear cuadernos de Jupyter en línea, que se almacenan en Google Drive, y ejecutarlos en un servidor remoto haciendo uso del backend de Google Compute Engine. La principal ventaja de hacer uso de este servicio en lugar de correr el código en una máquina en local es la disponibilidad gratuita, aunque limitada, de GPUs. Incluso si tu máquina cuenta con una GPU adecuada para las tareas que quieras realizar, la creación de entornos locales y la instalación de múltiples librerías necesarias pueden ser un buen motivo para optar por un servicio en línea, con bibliotecas preinstaladas como NumPy, SciPy, Pandas o Tensorflow, entre muchas otras.

Sin embargo, el uso de Google Colab trae consigo ciertos inconvenientes a tener en cuenta. El primero de ellos es la limitación de uso de GPU; si bien es cierto que la capacidad de la GPU asignada suele ser suficiente para muchas de las tareas que se llevan a cabo, su uso limitado por tiempo puede hacer que descartemos esta opción si vamos a entrenar modelos que requieran un largo tiempo de entrenamiento, o si necesitamos una disponibilidad mayor de tiempo por cualquier otro motivo. Además, dado que el código corre en un servidor remoto, para tener acceso a los ficheros o los datos asociados, es necesario que estos estén alojados en Google Drive o subirlos a un entorno temporal cada vez que quieras ejecutarlo.

Aunque algunos de estos inconvenientes se pueden solventar con una versión de pago de Google Colab, puede no ser suficiente en algunos casos concretos. Aun así, para el desarrollo de este proyecto esta herramienta ha sido de gran utilidad.

Capítulo 3

Adquisición y procesado de las señales

Para conseguir un clasificador capaz de distinguir entre el sonido radiado a través de un nudo, de aquel que procede directo de una larva, es necesario capturar las señales, así como realizar un procesamiento para adecuar las señales al entrenamiento del algoritmo. Esta segunda parte es la que más decisiones involucra, puesto que al analizar las señales capturadas en bruto surgen varias formas de procesarlas, cada una de ellas con sus ventajas y desventajas. Sin embargo, es una parte muy importante para el posterior desarrollo de un algoritmo de DL contar con una buena base de datos, con datos representativos de la realidad, que permita a la red neuronal encontrar patrones que diferencien a nuestras dos clases de interés.

En este capítulo se detalla la metodología seguida para conseguir los datos que se emplearán como entrada para el entrenamiento del modelo de DL.

3.1. Adquisición de señales

El sistema de adquisición usado consta de tres elementos: un array acústico de micrófonos MEMS (*Micro-Electro-Mechanical Systems*) y un sistema de adquisición y preprocesamiento basado en FPGA (*Field Programable Gate Array*).

3.1.1. Array acústico

Para capturar las señales se ha empleado un array planar [1], con sus elementos distribuidos en una superficie completamente plana, lo que da como resultado una respuesta con un diagrama de radiación bidimensional. En concreto, el array utilizado se muestra en la Figura 3.1 y consta de 486 micrófonos digitales MEMS del fabricante Knowles [46], con una apertura espacial de 35 cm en ambas dimensiones. Por su parte, la resolución angular del array 2D usado depende esencialmente de la frecuencia de trabajo y del ángulo de apuntamiento, reduciéndose el ancho del haz a medida que aumenta la frecuencia de trabajo [1], que para esta investigación es el rango comprendido entre 0 y 23.5 kHz.

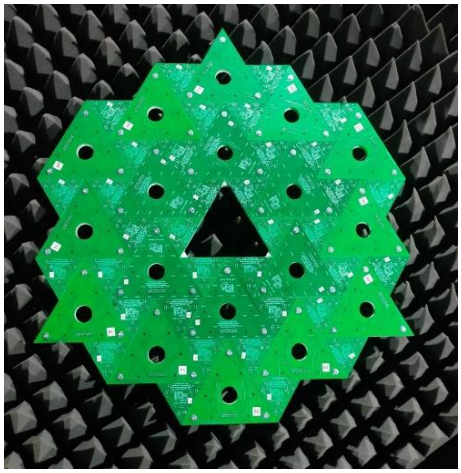


Figura 3.1: Array de micrófonos MEMS

También ocurre que el ancho del haz se ensancha a medida que aumenta el ángulo de apuntamiento desde la línea de mira, es decir, apuntando a 0°, hasta la máxima excursión de apuntamiento definida, es este caso de 50°. Cuanto menor es el ancho del haz, mejor es la resolución angular y más precisa es la determinación de la posición de las larvas en los haces bajo prueba, así como la posición de dos larvas próximas [1]. En estas condiciones, se define una cuadrícula de 61 por 61 puntos para acotar las vigas, que ocupan 120 por 120 cm como se detalla en la Figura 3.2, de esta forma obtenemos una resolución espacial de aproximadamente 2 cm.

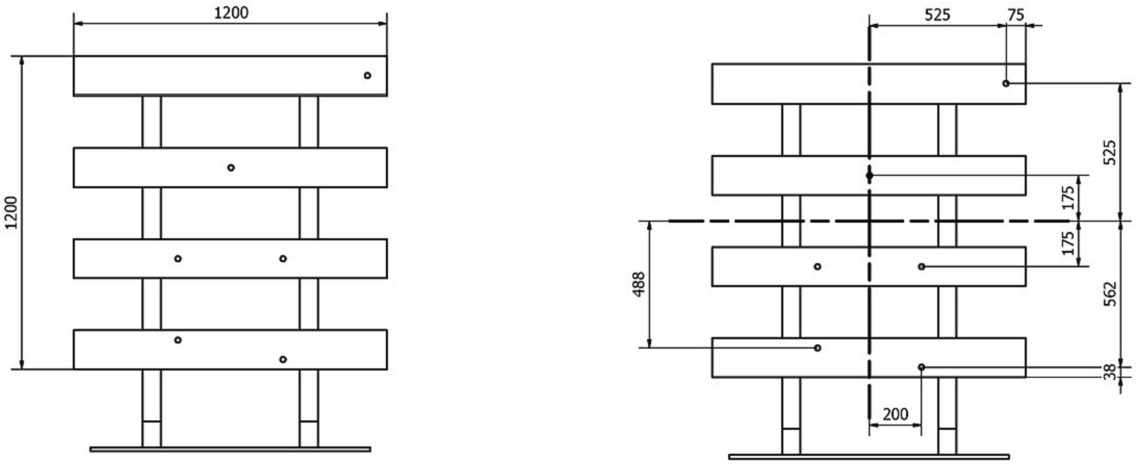


Figura 3.2: Tamaño de las vigas y posiciones de implantación de las larvas expresadas en milímetros. [1]

Dado que el array se colocó a una distancia de 60 cm de las vigas de madera en las que estaban colocadas las larvas, no se reúnen las condiciones necesarias para considerar la aproximación de onda plana, puesto que el array no está lo suficientemente alejado para asumir una situación de campo lejano. Típicamente el conformador *Delay and Sum* se basa en la premisa de campo lejano, de forma que el retraso con el que la señal llega a cada micrófono depende únicamente de la posición del sensor y del ángulo de apuntamiento [1]. Aun así, es posible adaptar este algoritmo a un escenario de campo cercano tomando en consideración que el retraso entre las señales dependerá también de la distancia relativa desde el emisor a cada sensor del array asumiendo una propagación esférica.

Puesto que en este caso el plano en el que se generan las emisiones acústicas es conocido, el retraso asociado a la propagación de la señal entre cada uno de los puntos de la cuadrícula definida se pueden determinar usando el algoritmo Delay and Sum, de forma que la salida del array se puede expresar según

$$y(\vec{r_g}, t) = \frac{1}{M} \sum_{n=1}^{N} w_n \cdot x_n(t - \tau_n(\vec{r_g})) \tag{3.1}$$

donde $\vec{r_g}$ representa la distancia entre el punto de referencia con cada uno de los puntos de interés definidos por la cuadrícula, N es el número de sensores que en este caso son 486, w_n es el peso aplicado al canal n del array (que en este caso vale siempre 1), y $x_n(t)$ representa la señal adquirida por el micrófono n. Por último, $\tau_n(\vec{r_g})$ indica el retraso temporal del sensor n del array al punto de referencia, considerando que la señal es una onda esférica, y se obtiene según

$$\tau_n(\vec{r_g}) = \frac{|\vec{r_g}| - |\vec{r_g} - \vec{r_n}|}{v_{\text{sonido}}} \tag{3.2}$$

siendo $\vec{r_n}$ el vector de distancias desde el punto de referencia al micrófono n del array.

3.1.2. Sistema de adquisición

El sistema de adquisición base empleado es el controlador sbRIO 9607 [47] mostrado en la Figura 3.3. Este dispositivo del fabricante National Instruments perteneciente a la familia de dispositivos Reconfigurable Input-Output (RIO) es un controlador integrado, que ejecuta NI Linux Real-Time con un FPGA Zynq-7020 y un procesador dual-Core 667 MHz. La FPGA tiene 96 entradas/salidas digitales, 81 de las cuales son usadas para la conexión con 162 micrófonos MEMS del array, de tal forma que en cada entrada/salida se multiplexan 2 micrófonos, mientras que las otras líneas se utilizan para generar el reloj y para el sincronismo. Para poder capturar las señales de los 486 sensores, se emplearon 3 tarjetas interconectadas.

Figura 3.3: Controlador sbRIO 9607 de National Instruments.

Por último, desde una aplicación PC desarrollada en LabVIEW 2021 se controlaban las operaciones de captura de las 3 tarjetas de adquisición de forma sincronizada.

3.1.3. Condiciones de adquisición

Las condiciones físicas de adquisición son importantes para el posterior análisis de resultados. En cuanto a las larvas implantadas, son 6 ejemplares de larva de la especie *Hylotrupes*

bajulus L. con pesos entre 0.22 gramos (g) y 0.35 g las cuales fueron depositadas en 4 vigas de madera de *Pinus sylvestris L.* de medidas 90x140x1200 mm previamente acondicionadas a un contenido de 12 % de humedad. Las larvas fueron extraídas de maderas demolidas de edificios históricos de la ciudad de Valladolid, España.

Para la implantación de las larvas, se hizo un agujero en el lado opuesto de cada viga, de tal forma que el agujero quedara a 10 mm del lado de observación. La larva se coloca al final del agujero, el cual se sella con papel tisú. Las larvas se distribuyeron de tal manera que se pudieran estudiar diferentes fenómenos de propagación del sonido en la madera, efecto de borde, etc.

Una vez implantadas las larvas en las muestras de madera, la posición inicial de las larvas se marca en las correspondientes partes de madera con una pegatina amarilla. Este montaje permite simular la presencia de varias infestaciones simultáneas. Las larvas se dejaron aclimatar durante 30 días tras los cuales se iniciaron las pruebas de escucha.

Los experimentos se realizaron en el interior de una cámara anecoica con el sistema de escucha paralelo al plano de las vigas y centrado respecto a las mismas a una distancia de 60 cm. La Figura 3.4 muestra una imagen del escenario de adquisición. En ella se detallan con una B cada viga (en inglés, *beam*) de madera, y con los números se indexa cada larva implantada.

(a) Vigas con las larvas implantadas. (b) Posición del array frente a las vigas.

Figura 3.4: Escenario de adquisición.

Respecto a las maderas, es importante para este estudio resaltar la presencia de los nudos en cada una de las vigas. En la Figura 3.5 se detalla mediante un sombreado naranja la localización de los nudos presentes en las distintas vigas. Esto es importante para el posterior análisis, en el que se encontrará una presencia significativa de actividad de uno de los nudos de la viga 1, y del nudo situado a la derecha de la viga 2.

Figura 3.5: Nudos presentes en las vigas bajo estudio.

3.2. Segmentación de señales

Las señales capturadas contienen 1108 muestras, que adquiridas a una tasa de muestreo de 50.000 muestras por segundo, equivale a 22.16 milisegundos (ms). Durante este tiempo es posible que se capturen varias emisiones acústicas distintas, que típicamente son de 1 ms de duración [1].

Para detectar automáticamente la presencia o no de una emisión acústica en una posición determinada, se hace el cálculo de la energía deslizante en base al siguiente algoritmo. Primero se digitaliza la señal $x_k[n]$ capturada por cada sensor k según

$$x_k[n] = x_k(nT), \qquad T = 1/f_s, \;\; n = 0, ..., N-1, \;\; k = 0, ..., K-1 \tag{3.3}$$

Cada señal digitalizada se pasa por un filtro digital anti-aliasing con una frecuencia de corte de 24.5 kHz y después por un filtro FIR paso banda $h[n]$ con banda de paso 1 kHz a 23.5 kHz y usando la técnica de la ventana, con una ventana de Hanning. Este filtrado se hace con el objetivo de eliminar la componente continua y filtrar en la banda de interés. De forma que la señal filtrada $y_k[n]$ se puede expresar como

$$y_k[n] = x_k[n] * h[n] \tag{3.4}$$

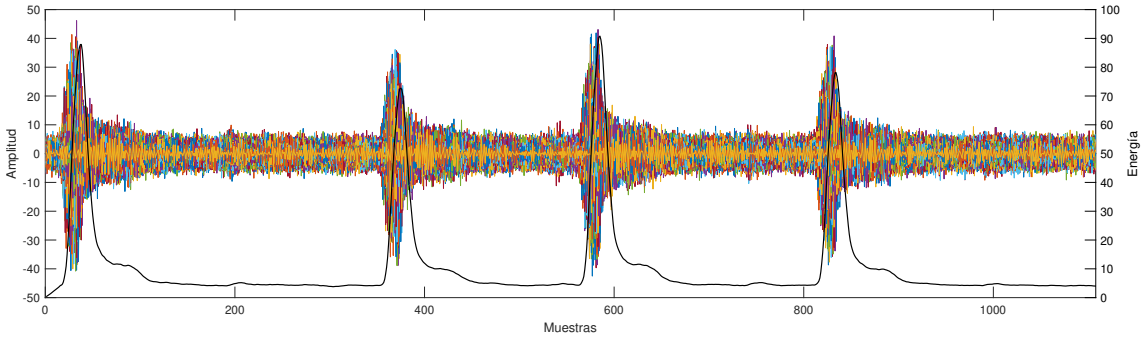
Una vez la señal es filtrada, para detectar si hay actividad de alguna de las larvas en la señal capturada, se calcula la energía deslizante $E_k[n]$ sobre una ventana de 2 ms para cada una de las señales $y_k[n]$ según

$$E_k[n] = \sum_{r=0}^{N-1} y_k^2[r] \cdot w^2[r-n], \qquad w[n] = 10 \leq n \leq M-1, \quad M = \frac{2 \cdot 10^{-3}\ \text{s}}{T} \tag{3.5}$$

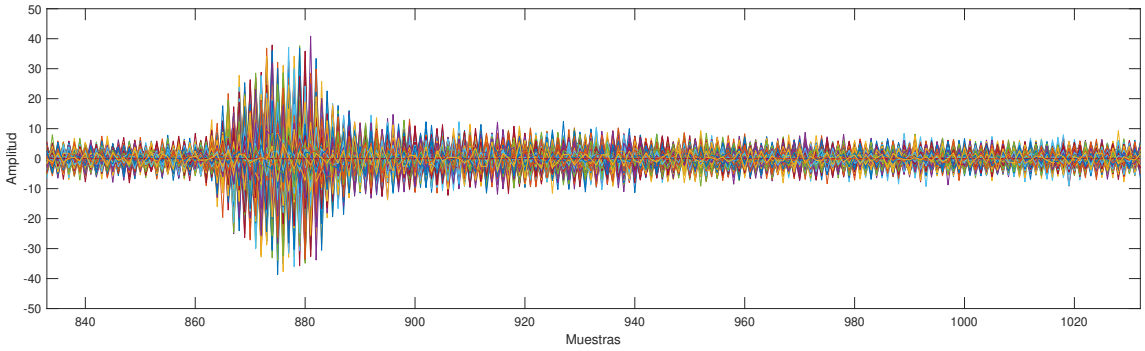
Posteriormente, se calcula el estimador de la energía media como el promedio de cada secuencia $E_k[n]$ como

$$E_{\text{media}}[n] = \text{media}\{E_k[n]\} \tag{3.6}$$

El estimador $E_{\text{media}}[n]$ se compara con un umbral, y en caso de que lo supere, se considera que hay actividad de una larva y se segmenta esa señal. En [1] se utilizó un criterio de segmentación en el que de cada captura de 22 ms sólo se segmentaba el fragmento de señal en el entorno donde se detectaba el pico con la máxima energía. Sin embargo, en algunas capturas es posible que en el tiempo que dura la muestra, diferentes larvas produjeran un sonido. Es el caso representado en la Figura 3.6(a), en la cual se muestran las 486 señales capturadas por cada uno de los micrófonos que componen el array junto con el cálculo resultante de la energía deslizante sobre el promedio de esas señales. Se observa que hay cuatro picos de energía claramente separados, correspondientes a distintos sonidos. Extrayendo más de un segmento por señal, obtenemos un número considerablemente superior de muestras, necesario para el entrenamiento de un modelo de DL.



(a) Señales de los 486 micrófonos con su energía deslizante promedio.



(b) Segmento extraído.

Figura 3.6: Promedio de las 486 señales originales con su energía deslizante y un segmento extraído.

Es por eso que se desarrolló un algoritmo de segmentación multiblanco, para el cual fue necesario tomar en consideración determinados aspectos para verificar si la señal segmentada era válida o no. Primero, fue necesario tener en cuenta que los picos de la energía deslizante no estuvieran muy pegados al inicio o al final de la captura, puesto que si se detecta el pico a unas pocas muestras del comienzo habremos perdido gran parte de la información anterior, y si se se detecta muy cerca de la última muestra perderemos parte de la información posterior. Es por eso, que se optó por considerar unos valores de guarda de 1 ms, o lo que es lo mismo, 50 muestras.

Para comprobar que la señal segmentada fuera válida se buscó el máximo de la energía promediada, a partir de ese valor se consideraron las 100 muestras posteriores, lo que equivale a 2 ms de señal. De estás 100 muestras se validó que hubiera al menos 50 muestras a cada lado, con lo que se obtuvieron señales de 200 muestras, o 4 ms. En la Figura 3.6(b) se muestran las señales extraídas del pico de energía de la Figura 3.6(a) situado en torno a la muestra 800.

Elegir adecuadamente estos valores de guarda es importante para el posterior conformado, ya que si no hay muestras suficientes para retrasar o adelantar en el proceso del *Delay and Sum* se pueden cometer fallos que afecten a la señal conformada.

Gracias a esta segmentación multiblanco conseguimos aumentar significativamente el número de señales adquiridas. En la Figura 3.7 se muestra el número de segmentos extraídos por cada señal para el caso de la segmentación multiblanco, y su comparación con la segmentación monoblanco. Con esto pasamos de unas 58000 señales a 148000, un tamaño más adecuado para tareas de aprendizaje profundo.

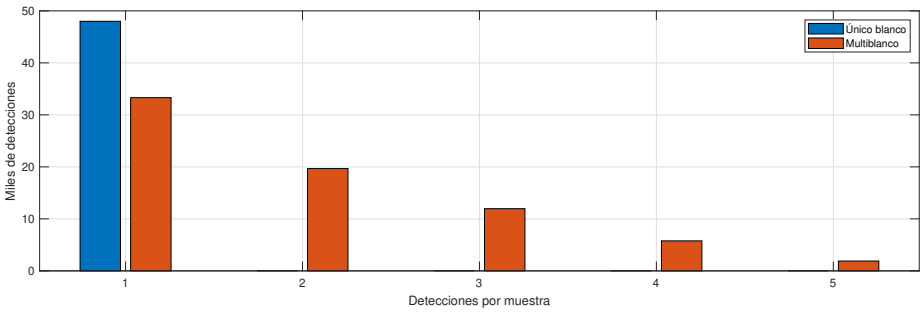


Figura 3.7: Número total de detecciones por captura con diferentes valores de guarda.

3.3. Conformación de las señales

Una vez segmentadas las señales, para obtener un sólo haz apuntando en la dirección de la que provenga el sonido es necesario realizar una conformación. En este caso se realizará un conformado *Delay and Sum* explicado en la Sección 2.1.1 que nos permite obtener una imagen acústica como la mostrada en la Figura 3.8. Esta imagen representa, según la escala de color, el nivel de energía de la señal correspondiente a un haz apuntando en cada dirección de la cuadrícula especificada.

Aunque la información de interés en este caso queda contenida en un plano 2D, con las coordenadas X e Y, el procesado realizado nos permite también conocer una tercera dimensión que representa el rango o profundidad, y una cuarta, en función de la frecuencia

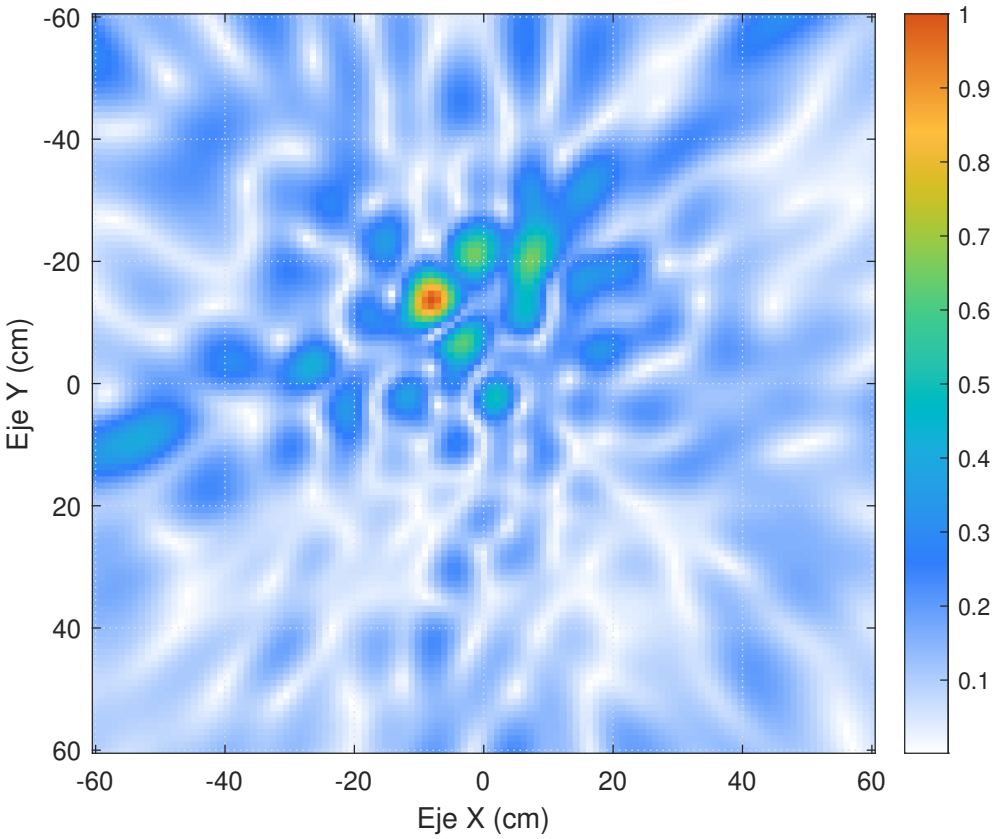


Figura 3.8: Imagen acústica

de trabajo del array.

Para realizar esta tarea, el Grupo de Procesado en Array (GPA) de la Universidad de Valladolid (UVa) ha desarrollado una serie de librerías para LabVIEW que facilitan significativamente la tarea.

3.4. Etiquetado de los datos

Una vez realizada la conformación, obtenemos la posición 2D (azimut y elevación) del máximo. En base a esta información se ha realizado una agrupación de las señales conformadas en función de su localización, basada en la información conocida de la posición original de las larvas y los nudos. En la Tabla 3.1 se detalla las posiciones exactas en la que se implantaron las larvas, así como aquellas en las que se detectó la presencia de los nudos que afectan principalmente a los resultados obtenidos.

En la Figura 3.9 se muestran la posiciones de cada una de las detecciones con una escala de color que indica la frecuencia de detección en esa coordenada. La cruz naranja representa la posición teórica del objetivo, sin embargo, esta medida no debe tomarse como una referencia exacta puesto que a lo largo de los experimentos tanto el array como las vigas de madera pudieron sufrir ligeros desplazamientos, además de los errores que pudieran

Blanco	ID	Posición (x, y)	Blanco	ID	Posición (x, y)
Larva 1	1	(525, 525)	Larva 5	5	(-200, -488)
Larva 2	2	(0, 175)	Larva 6	6	(200, -562)
Larva 3	3	(-200, -175)	Nudo 1	7	(80, 525)
Larva 4	4	(200, -175)	Nudo 2	8	(525, 175)

Tabla 3.1: Posiciones de interés dadas en milímetros.

cometerse durante la calibración. Aún así es notable el caso de la larva 2 o la 3, en los que su posición teórica dista considerablemente de la localización de la mayoría de detecciones.

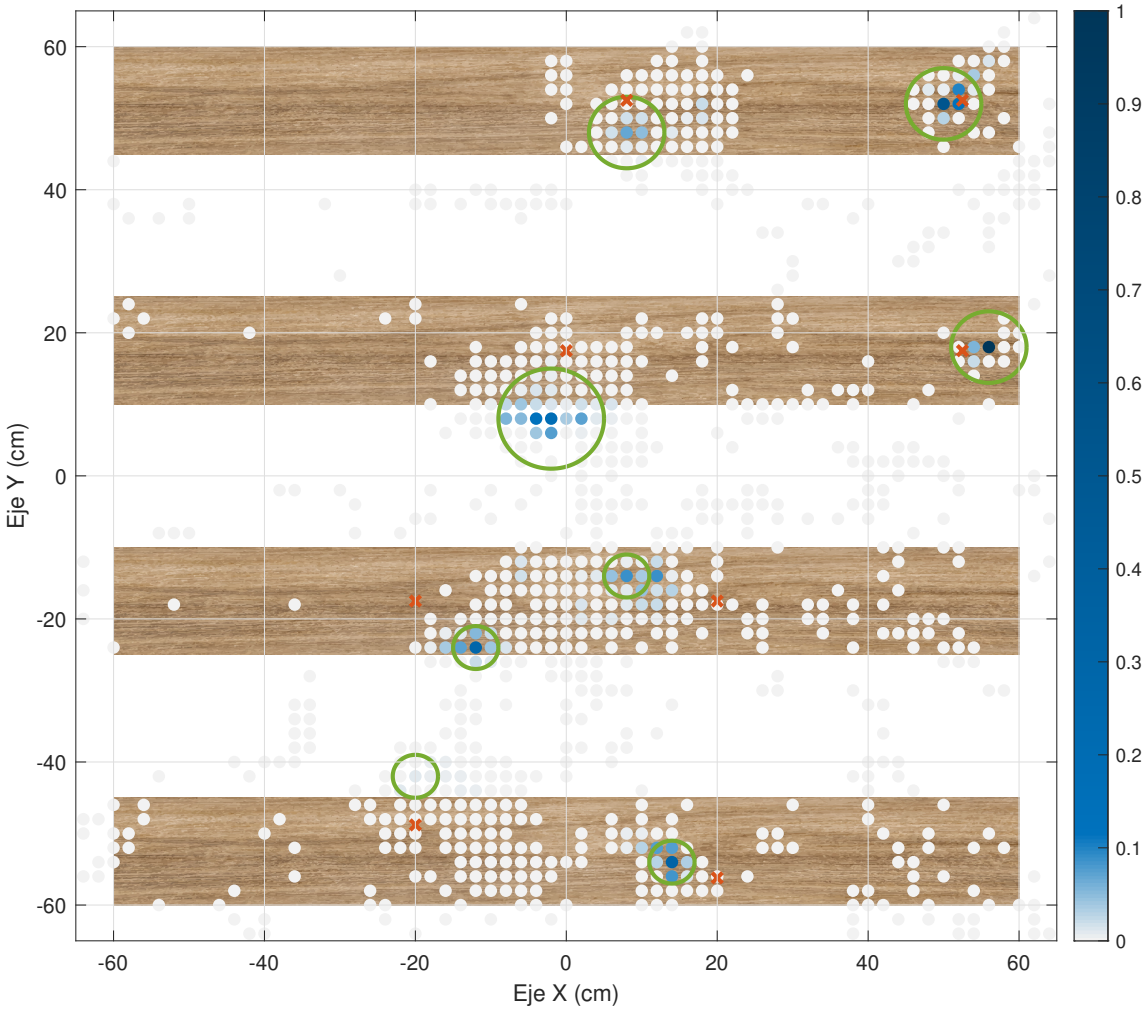


Figura 3.9: Posiciones de las señales capturadas.

El proceso de etiquetado de las señales capturadas está basado en las coordenadas del máximo detectado en la imagen acústica de las señales conformadas. Puesto que se trata de un etiquetado automático cuenta con ventajas, ya que no es necesario contar con un experto que realice el etiquetado manual disminuyendo el tiempo y esfuerzo asociado a dicha tarea, pero presenta una serie de inconvenientes como la incertidumbre asociada a la etiqueta, ya que no se está aplicando ninguna técnica adicional para validar que el

etiquetado es correcto.

Es por eso que para acotar lo máximo posible la zona de detecciones de cada blanco, se aplica una región alrededor de los puntos donde se ha observado una mayor frecuencia de detección, por lo que cabría esperar que la mayoría de esos sonidos correspondan a la larva que se implantó en ese lugar. Para realizar este "filtrado" basado en la posición, se eligió un radio específico en función de la dispersión de los datos recabados en cada zona. Las circunferencias verdes mostradas en la Figura 3.9 indican el radio dentro del cuál las señales se etiquetan como larva, nudo o ruido. Por ejemplo, el radio usado para etiquetar la larva 3 es mayor que el resto, puesto que el número de detecciones es elevado en un entorno mayor que en otros casos.

Por otra parte, las larvas 2 y 5 muestran un mayor nivel de actividad en posiciones alejadas de la localización original. De hecho, la circunferencia elegida para seleccionar los sonidos de la larva 5 se encuentra incluso fuera de la madera. Esto se explica debido a que estas larvas fueron implantadas en una cara perpendicular al punto de observación. [1]

Finalmente, tras este etiquetado, de las 148.000 señales capturadas, 125.450 entran dentro de las regiones marcadas y por tanto son consideradas larva o nudo y empleadas para entrenar los posteriores modelos.

3.5. Análisis de los datos

Ya con los datos preparados, se realizó un análisis para estudiar la actividad de las larvas con el objetivo de conocer el problema abordado y conseguir una mejor interpretación de los datos.

Un aspecto a estudiar interesante es ver evolución de la posición del blanco a lo largo del experimento. En función de qué parte de la madera esté comiendo la larva, la localización del sonido emitido puede variar. Si la larva está alimentándose de una parte de la madera de frente a una fibra que termine en nudo, es muy probable que el sonido salga principalmente por ese nudo puesto que la propagación será más favorable, con un nivel de atenuación menor.

En la Figura 3.10 se representó la evolución de la posición de las señales capturadas a lo largo del tiempo, con una normalización para cada blanco. Puesto que los valores de posición obtenidos por el array tienen una resolución espacial de 2 cm, y el conjunto de datos es bastante grande, se optó por promediar la posición en X y promediar la posición en Y de cada 50 muestras consecutivas de un mismo blanco. De esta forma, cada punto del gráfico representa la posición promedio a lo largo de 50 capturas.

Con esto, conseguimos ver que en algunos casos, como el del nudo 2, la variación de la posición es mínima a lo largo del tiempo, pero en otros casos como la larva 1, se puede percibir la trayectoria que siguió la misma.

Por su parte, si queremos comparar la relación entre las detecciones entre unos blancos y otros, podemos ver conjuntamente el número de detecciones por día de cada uno de ellos. Es lo que se muestra en la Figura 3.11, donde se normaliza entre cero y uno el número de detecciones de cada día entre el 22 de diciembre de 2021 y el 4 de febrero de 2022, fechas en las que se realizaron las escuchas. Con esto, vemos que la larva 1 y su nudo asociado, tienen una correlación marcada en lo que a las detecciones se refiere, mientras

que la larva 2 y su respectivo nudo no presentan una aparición conjunta muy marcada, más bien cuándo se detectan muchas ocurrencias de uno se detectan pocas del otro.

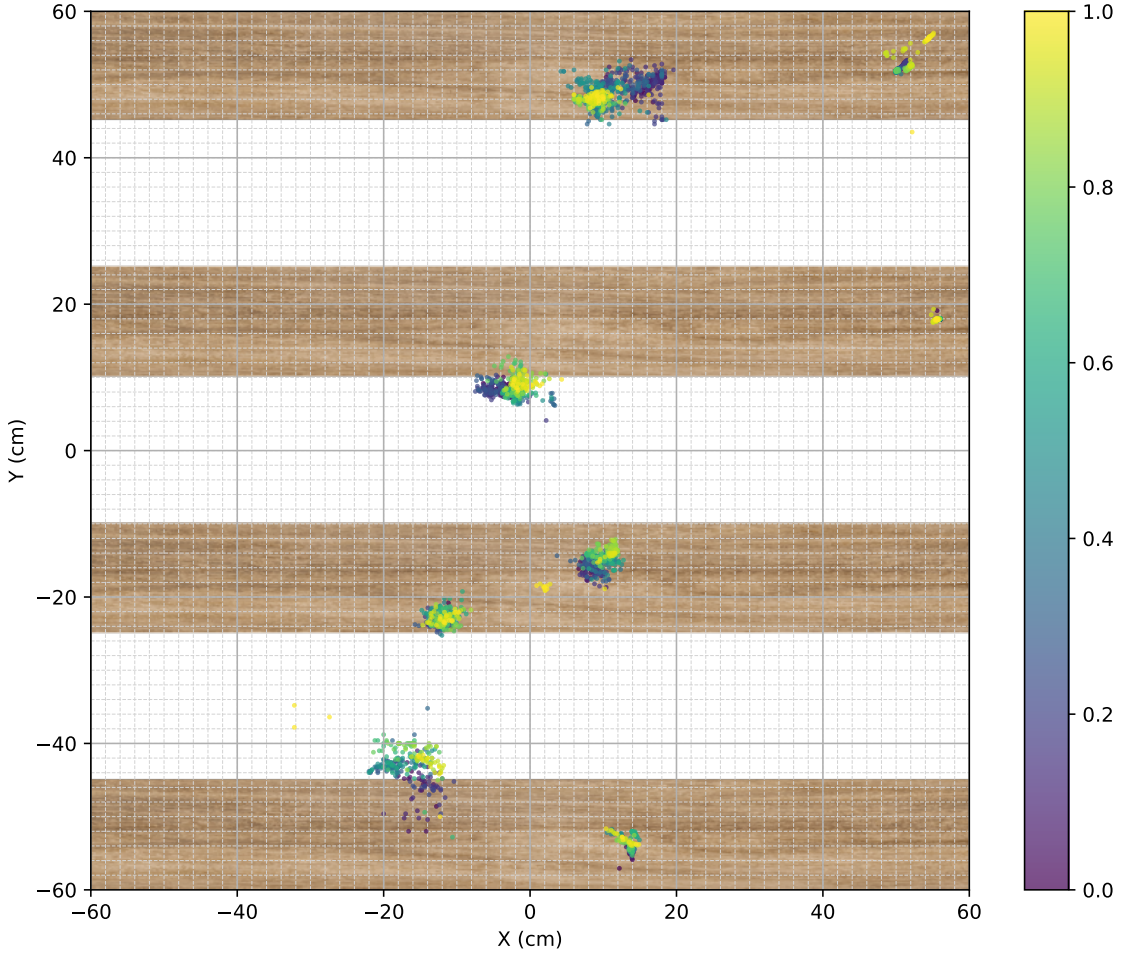


Figura 3.10: Evolución temporal de la posición.

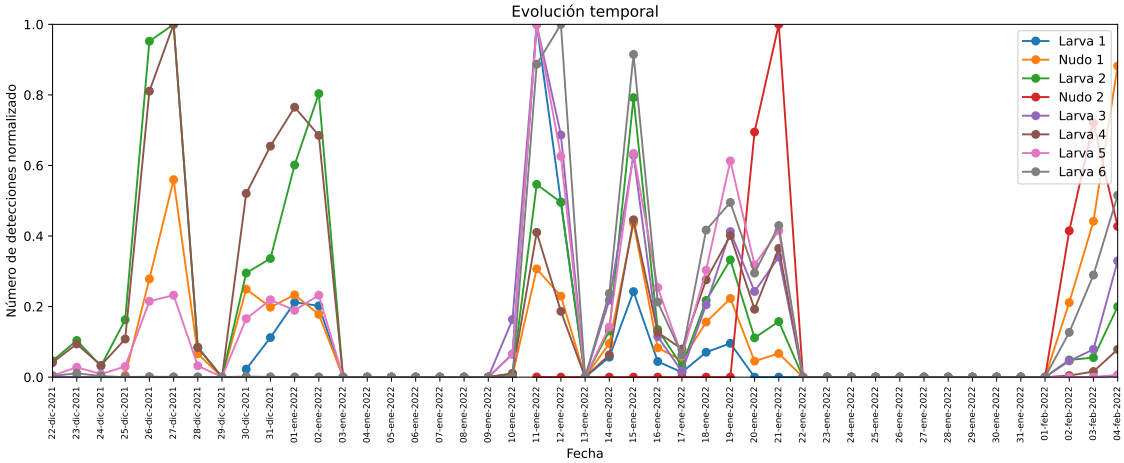


Figura 3.11: Evolución temporal de las detecciones de cada blanco.

3.6. Preparación de los datos

Existen diversas formas de introducir los datos a una red neuronal, por tanto elegir cómo tratar los datos antes de pasárselos al algoritmo es una parte importante. Depende de la arquitectura empleada, para una CNN es necesario pasar imágenes, o datos en forma matricial, mientras que para un MLP se requiere una secuencia de datos unidimensional.

3.6.1. Escalogramas

En este caso, puesto que la idea inicial era probar el enfoque de *transfer learning* a partir de algún modelo preentrenado con un propósito similar, se pensó en crear imágenes 2D que recogieran una representación tiempo-frecuencia, conservando tanto las características temporales como las frecuenciales con herramientas como la Transformada Corta de Fourier (STFT) para crear espectrogramas o la Transformada Wavelet para crear escalogramas.

Debido a la longitud tan pequeña de las señales segmentadas, la STFT presentaba el inconveniente de perder mucha resolución frecuencial, es por eso que se optó por crear escalogramas. Para ello, se hizo uso del paquete de MATLAB Wavelet Toolbox™.

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} f(t)\ \psi\left(\frac{t-\tau}{s}\right) dt \tag{3.7}$$

donde s es la escala de la wavelet, τ el retardo, $f(t)$ la señal original y $\psi(t)$ es una función conocida como wavelet madre, que en este caso ha sido la función Morse [48].

La Transformada Wavelet continua, descrita en la Ecuación (3.7), permite un análisis tiempo-frecuencia multiresolución [49], con una mayor resolución frecuencial en bajas frecuencias, y una mayor resolución temporal para las altas frecuencias.

En la Figura 3.12 se muestra un ejemplo del cálculo de un escalograma, en el que para cada instante temporal hay un espectro en frecuencia.

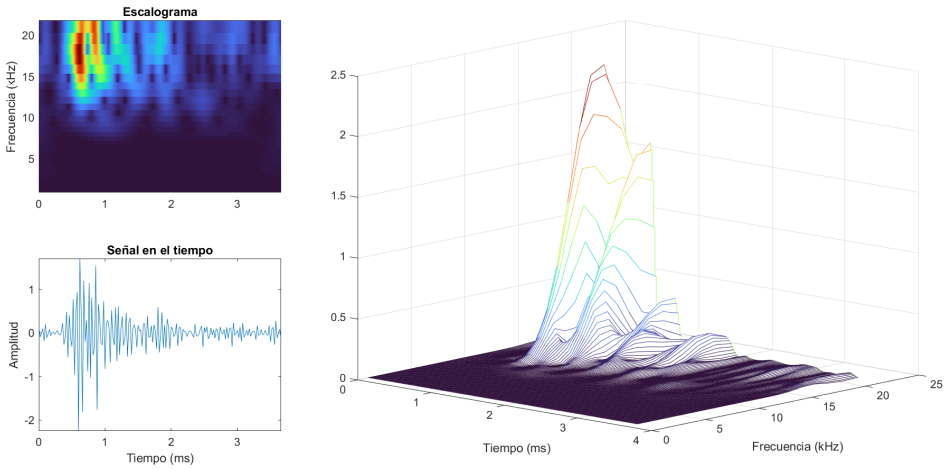


Figura 3.12: Escalograma obtenidos.

3.6.2. Señal en tiempo

Otra opción es que la entrada al modelo sea directamente las muestras de la señales en el tiempo. Esta opción requiere de un entrenamiento desde cero y la creación de una red neuronal específica para el caso. Las señales están almacenadas en formato CSV (del inglés, *comma-separated values*) y etiquetadas como se muestra en la Figura 3.13.

	Rango	X	Y	Frecuencia	0	1	2	3	4	5	...	175	176	177	178	179	180	181	182	183	Etiqueta
0	40.0	50.0	52.0	20500.0	0.270858	-0.249036	0.209702	-0.055706	-0.129341	0.230508	...	-0.116690	0.013951	0.002817	-0.054948	0.195177	-0.246405	0.116132	0.123058	-0.258847	1
1	4.0	50.0	52.0	20500.0	-0.222247	0.051623	0.228247	-0.545894	0.748431	-0.619971	...	-0.917756	0.661228	-0.107808	-0.302339	0.556274	-0.702506	0.622949	-0.202278	-0.239497	1
2	32.0	50.0	52.0	20500.0	0.072546	0.192326	-0.481111	0.657858	-0.551835	0.140797	...	0.268780	-0.391941	0.421407	-0.223014	-0.088699	0.330080	-0.459706	0.455869	-0.313085	1
3	8.0	52.0	54.0	20000.0	0.090044	0.241708	-0.350381	0.379089	-0.279570	0.172295	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1
4	0.0	52.0	54.0	20500.0	1.341920	-0.978519	0.349570	0.318965	-0.929788	1.300298	...	0.000703	0.052869	-0.108753	0.165380	-0.193274	0.101556	0.049479	-0.091145	0.114016	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
125445	24.0	56.0	18.0	20500.0	0.123691	-0.234821	0.134918	0.038254	-0.326047	0.543598	...	-0.521056	0.306667	0.019673	-0.351852	0.598964	-0.727436	0.642033	-0.399464	0.103897	8
125446	16.0	56.0	18.0	20500.0	0.044943	-0.025161	-0.077661	0.050448	0.086206	-0.226170	...	-0.320187	0.204169	0.168463	-0.627714	0.949822	-0.901983	0.600997	-0.226676	-0.105733	8
125447	8.0	56.0	18.0	20500.0	0.075704	0.017430	-0.011758	-0.168536	0.224930	-0.095919	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	8
125448	20.0	56.0	18.0	20500.0	-0.049293	0.099470	-0.145217	0.091989	0.154123	-0.347539	...	0.641796	-0.665461	0.561219	-0.271931	-0.139775	0.614828	-0.993523	1.137212	-0.780617	8
125449	8.0	54.0	18.0	20500.0	-0.650898	0.521502	-0.229354	-0.111657	0.367706	-0.451028	...	-0.148481	-0.157575	0.373894	-0.412516	0.196335	0.089286	-0.232932	0.207993	-0.157597	8

125450 rows × 189 columns

Figura 3.13: Estructura de la base de datos con las señales en tiempo

La estructura es la siguiente:

- `Rango`. Indica la profundidad de la señal capturada.
- `X`. Indica la coordenada X de la señal capturada.
- `Y`. Indica la coordenada Y de la señal capturada.
- `Frecuencia`. Indica la frecuencia de trabajo del array para la señal capturada.
- `0 - 183`. Cada uno de los valores corresponde con una de las muestras de la señal temporal.
- `Etiqueta`. Indica la etiqueta de la señal en cuestión, siguiendo el ID mostrado en la Tabla 3.1.

3.6.3. Espectro en frecuencia

Por último, se optó también por probar una representación unidimensional, en este caso en frecuencia. Para ello, se calculó una Transformada Discreta de Fourier (DFT) de 46 puntos entre 1 kHz y 23.5 kHz, con una separación frecuencial de 500 Hz.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j(2\pi/N)} \tag{3.8}$$

donde k representa los índices de frecuencias y N es el número de muestras de la señal $x[n]$. [50]

La DFT, cuya expresión se detalla en la Ecuación 3.8, es una secuencia que corresponde a muestras equiespaciadas en frecuencia de la transformada de Fourier en tiempo discreto de la señal. La DFT tiene un papel crucial en el procesamiento de señales, en parte porque existen algoritmos eficientes para su cálculo. [50]

La estructura del CSV es similar a la de la señal temporal, y se detalla a continuación.

- `Rango`. Indica la profundidad de la señal capturada.
- `X`. Indica la coordenada X de la señal capturada.
- `Y`. Indica la coordenada Y de la señal capturada.
- `Frecuencia`. Indica la frecuencia de trabajo del array para la señal capturada.
- `1000 - 23500`. Cada uno de los valores de las muestras de la DFT.
- `Etiqueta`. Indica la etiqueta de la señal en cuestión, siguiendo el ID mostrado en la Tabla 3.1.

Normalización de la respuesta frecuencial

La respuesta en frecuencia de cada micrófono, mostrada en la Figura 3.14, representa su sensibilidad a cada frecuencia. Se puede observar que ente 3 y 15 kHz presenta una zona aproximadamente plana, pero su sensibilidad aumenta en torno a 22 kHz, donde se encuentra una resonancia de los micrófonos [46]. También se encuentra un aumento de esta sensibilidad en las frecuencias más bajas.

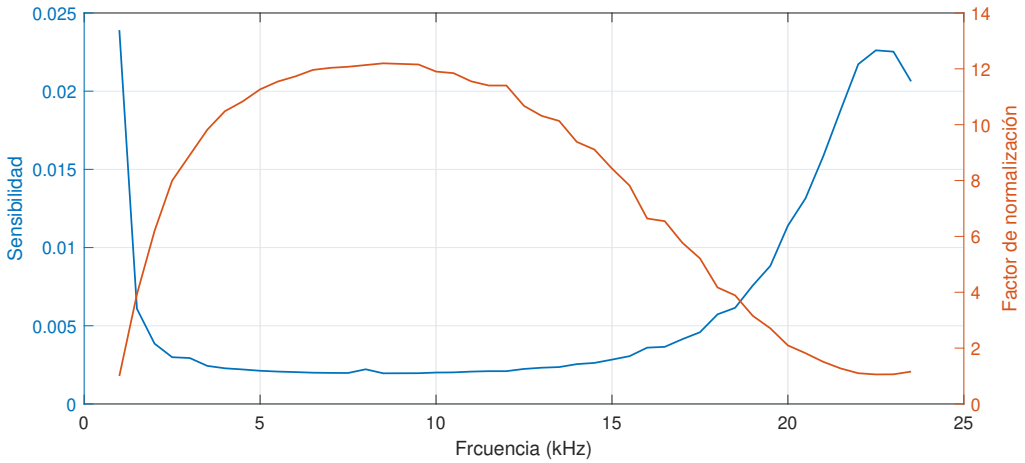


Figura 3.14: Caracterización de la respuesta en frecuencia del array.

Puesto que en estudios previos [2] se han empleado las frecuencias más bajas (en torno a 10 kHz) para la identificación de sonidos producidos por larvas de insectos xilófagos, podría ser útil tratar de normalizar esta respuesta frecuencial, para que los algoritmo de DL no den una importancia mayor a aquellas frecuencias en torno a 22 kHz, donde los micrófonos empleados presentan esa resonancia. Por este motivo, se normalizó esta respuesta de manera que multiplicando la amplitud de la DFT por el coeficiente correspondiente de la línea naranja mostrada en la Figura 3.14 obtenemos una respuesta frecuencial plana en referencia al primer valor (1000 Hz), que es la frecuencia para la cual se obtiene el valor más alto.

Capítulo 4

Desarrollo del clasificador

A la hora de elegir una arquitectura de red neuronal se plantearon varias opciones para comparar su comportamiento y finalmente ver cuál presentaba un mejor resultado de clasificación para nuestros datos.

La primera opción planteada fue utilizar una arquitectura de redes convolucionales que recibieran como datos de entrada los escalogramas, puesto que contienen mucha más información que la señal cruda. Si esta primera opción arrojara resultados interesantes, se plantearía después utilizar un MLP cuya entrada sería las muestras de la señal en tiempo o el espectro en frecuencia obtenido de la DFT.

4.1. *Transfer learning*: Xception

Antes de diseñar una red neuronal compleja completa desde cero, en muchos casos es interesante buscar entre las redes existentes aquellas que tengan un propósito parecido y usen datos de entrenamiento similares a los de interés. Además, buscar una arquitectura estándar facilita el *transfer learning* [51], y permite obtener unos resultados buenos de una manera más simple y sin necesidad de preparar un conjunto de datos demasiado grande.

Es por esto que se pensó en este enfoque como primera opción. Para una primera aproximación se probó el algoritmo Xception [52], una red neuronal convolucional con una arquitectura de 71 capas pre-entrenada con más de un millón de imágenes de la base de datos de ImageNet, cuya adaptación a los escalogramas para nuestro problema de clasificación binaria se detalla a continuación.

Layer (type)	Output Shape	Param #	Trainable
input (InputLayer)	[(None, 150, 150, 3)]	0	Y
rescaling (Rescaling)	(None, 150, 150, 3)	0	Y
xception (Functional)	(None, 5, 5, 2048)	20861480	N
global_average_pooling2d	(None, 2048)	0	Y

```
 (GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0         Y

 dense (Dense)               (None, 1)                 2049      Y

=================================================================
Total params: 20,863,529
Trainable params: 2,049
Non-trainable params: 20,861,480
_________________________________________________________________
```

En primer lugar, una capa de entrada define un tamaño de imagen de 150x150 píxeles con 3 canales de color (RGB), que en este caso son 3 copias idénticas de una imagen en escala de grises, en este caso, nuestro escalograma. Después, una capa de reescalado convierte los valores entre 0 y 255, típicos de una imagen, a valores entre -1 y 1 requeridos por el modelo Xception. Una vez adecuada la entrada a este modelo, se usan todas sus capas pero sin ser entrenadas, para no modificar los pesos aprendidos por esta arquitectura en su entrenamiento con la base de datos ImageNet. A la salida de la red Xception, una capa de Pooling reduce a un vector de tamaño 2048 las características extraídas por la red, promediando en la dimensión espacial. Después, se incluye una capa de *dropout* que ayude a prevenir el sobreajuste durante el entrenamiento. Y por último, una capa densa con una sola neurona de salida permite obtener el resultado de la clasificación binaria.

Para entrenar este modelo se utilizaron los siguientes hiperparámetros. El modelo fue entrenado usando como método de optimización Adam. El entrenamiento se hizo durante 50 épocas con un tamaño de lote de 64 y una tasa de aprendizaje de 0.001. En cuanto a las funciones de activación de las neuronas, todas ellas utilizan ReLU salvo la última neurona, cuya salida determina a qué clase pertenece la imagen de la entrada, y puesto que se trata de un problema de clasificación binaria, se optó por usar una función sigmoide. Además, para este modelo, la tasa de *dropout* escogida para prevenir sobreajuste fue 0.2.

El conjunto de datos con el que fue entrenado el modelo se construyó utilizando la librería de Keras `image_dataset_from_directory`, mediante la cual se convirtieron las imágenes de los escalogramas, estructuradas en directorios, en un *dataset* de tipo `tf.data.Dataset`. Un ejemplo de imágenes que componen el *dataset* se muestra en la Figura 4.1.

Figura 4.1: Ejemplos de imágenes etiquetadas del conjunto de datos.

4.1.1. *Fine-tuning*

Como un paso adicional al *transfer learning* está el proceso llamado "fine-tuning", una técnica ampliamente utilizada que consiste en descongelar todo el modelo obtenido anteriormente y volver a entrenarlo con los nuevos datos, con una tasa de aprendizaje muy baja para evitar cambios drásticos. De este modo se pueden conseguir mejoras significativas, adaptando de forma incremental las características pre-entrenadas a los nuevos datos. [53]

En este caso, una vez entrenada únicamente la capa densa, sin modificar los pesos originales del modelo base Xception, se hizo un segundo entrenamiento en el que se mantuvieron todos los hiperparámetros, excepto la tasa de aprendizaje, que se redujo a un valor de 10^{-5}, y el número de épocas, que se realizaron 5. Con esto se consiguen modelos con un comportamiento más adecuado en la tarea de clasificación específica.

4.2. Red neuronal convolucional (CNN)

Tras comprobar que con una red convolucional pre-entrenada era posible distinguir entre nudo y larva, se pensó en crear un modelo más sencillo entrenado *ad hoc*, para comparar su rendimiento.

Para esto se diseñó la red neuronal convolucional de 14 capas detallada a continuación.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 25)	250
batch_normalization (BatchNormalization)	(None, 150, 150, 25)	100
max_pooling2d (MaxPooling2D)	(None, 75, 75, 25)	0
conv2d_1 (Conv2D)	(None, 75, 75, 50)	11300
dropout (Dropout)	(None, 75, 75, 50)	0
batch_normalization_1 (BatchNormalization)	(None, 75, 75, 50)	200
max_pooling2d_1 (MaxPooling2D)	(None, 38, 38, 50)	0
conv2d_2 (Conv2D)	(None, 38, 38, 75)	33825
batch_normalization_2 (BatchNormalization)	(None, 38, 38, 75)	300
max_pooling2d_2 (MaxPoolin	(None, 19, 19, 75)	0

```
 g2D)

 flatten (Flatten)           (None, 27075)             0

 dense (Dense)               (None, 512)               13862912

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 13909400 (53.06 MB)
Trainable params: 13909100 (53.06 MB)
Non-trainable params: 300 (1.17 KB)
_________________________________________________________________
```

La arquitectura de la red neuronal convolucional propuesta para procesar imágenes de 150x150 píxeles, que en este caso es de un solo canal, comienza con capas convolucionales que utilizan un tamaño de kernel de 3x3 píxeles. Estas capas están seguidas por capas de normalización, que estabilizan y aceleran el entrenamiento, y capas de *max pooling* con un tamaño de 2x2 píxeles para reducir la dimensionalidad. Este patrón de capa convolucional, normalización y *max pooling* se repite, disminuyendo gradualmente el tamaño de las imágenes y aumentando el número de filtros para capturar características más complejas. Finalmente, se obtienen 75 mapas de características de tamaño 19x19, que se aplanan mediante una capa Flatten, resultando en 27.075 valores. Estos se conectan a una capa densa de 512 neuronas y, por último, a una única neurona que determina la clase a la que pertenece la imagen.

El modelo fue entrenado según los siguientes hiperparámetros. Como método de optimización se empleó Adam, con un entrenamiento de 20 épocas y un tamaño de lote de 64. La tasa de aprendizaje se fijó en 0.001 y como función de activación de las neuronas se eligió ReLU salvo para la última neurona, que se utilizó la función sigmoide. Se utilizaron dos capas de *dropout*, con una tasa de 0.2 después de la segunda capa convolucional, y de 0.3 entre las dos capas densas. También se probó a entrenar el modelo eliminando estas capas de *dropout*. Por su parte, el conjunto de datos se construyó de manera idéntica al modelo anterior salvando las diferencias propias de utilizar imágenes de un solo canal.

4.3. Perceptrón multicapa (MLP)

Otra de las opciones planteadas fue la de entrenar un perceptrón multicapa con datos unidimensionales, o bien la señal en el dominio del tiempo, o bien su espectro en frecuencia a través de la DFT.

Para la elección del número de capas y el número de neuronas del modelo se han tenido en cuenta las recomendaciones mostradas en la Tabla 4.1.

Para el problema de clasificación que se pretende resolver se plantearon dos opciones, ambas con dos capas ocultas y con número de neuronas decrecientes. En la primera de ellas la primera capa contaría con 64 neuronas y 32 la segunda, mientras que en la otra

Número de capas ocultas	Resultado
Ninguna	Capaz de representar funciones separables de manera lineal
1	Permite aproximar cualquier función que contenga un mapeo continuo de un espacio finito a otro
2	Puede representar un límite de decisión arbitrario con precisión arbitraria con funciones de activación racionales y puede aproximar cualquier mapeo suave con cualquier precisión.

Tabla 4.1: Regla para determinar el número de capas ocultas. [54]

opción la primera tendría 32 y la segunda 16. Con esto podemos comparar el rendimiento de arquitecturas distintas en términos de complejidad.

A continuación se muestra la estructura en capas del modelo planteado para el caso de 64 neuronas en la primera capa y 32 en la segunda. Se trata de una arquitectura muy simple, con una carga computacional baja en relación a la mayoría de arquitecturas de DL por lo que puede ser entrenada en poco tiempo y sin necesidad de hacer uso de una GPU. Mientras que este modelo ocupa 20 *kilobytes* (KB) en parámetros, los modelos basados en arquitecturas convolucionales previamente mencionados tienen un peso del orden de decenas de *megabytes* (MB).

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense   (Dense)             (None, 64)                3008

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 5121 (20.00 KB)
Trainable params: 5121 (20.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

En este caso los datos de entrada son los 184 valores de cada muestra de la señal temporal o los 46 correspondientes a cada una de las frecuencias de la DFT.

4.4. Entrenamiento de las redes neuronales

Para el desarrollo y entrenamiento de los distintos clasificadores propuestos se ha hecho uso del entorno gratuito de "Google Colab"que permite el uso de una GPU NVIDIA® Tesla T4 con 15 GB de RAM que en el caso de los modelos probados ha sido suficiente. Gracias a esta herramienta se ha podido reducir notablemente el tiempo de entrenamiento

frente a haber entrenado los modelos en una máquina personal. La arquitectura de la GPU empleada se muestra en la Figura 4.2.

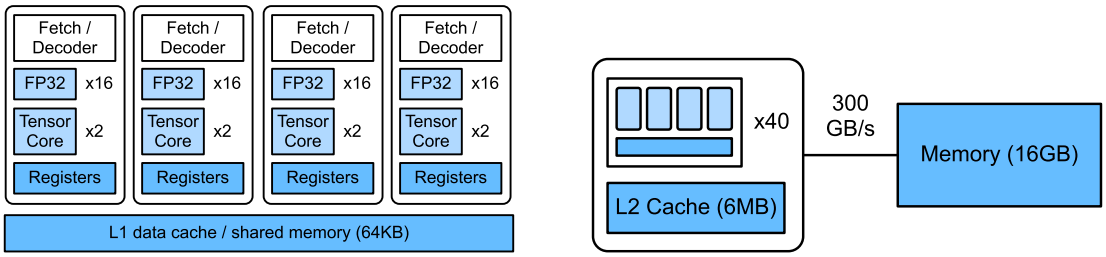


Figura 4.2: Arquitectura de la GPU NVIDIA® Tesla T4.

Capítulo 5

Resultados

En este capítulo se presentan los resultados conseguidos en el entrenamiento de las diferentes redes neuronales expuestas en el Capítulo 4. Se presentarán las curvas de entrenamiento de todos los modelos, así como los resultados obtenidos del mejor modelo de cada arquitectura a través de la matriz de confusión, y se evaluará el poder de clasificación de cada uno de ellos mediante el análisis de la curva ROC.

5.1. *Transfer learning*: Xception

Inicialmente se probó la arquitectura de CNN Xception, diseñada para la clasificación de imágenes. En la Figura 5.1 vemos las curvas de aprendizaje del modelo a medida que transcurría el entrenamiento. La línea gris punteada que se encuentra en torno a la época 10 indica el límite entre la primera fase (*transfer learning*) en la que los pesos del modelo base estaban congelados, y la segunda (*fine-tuning*) en la que sí se entrenaban estos pesos.

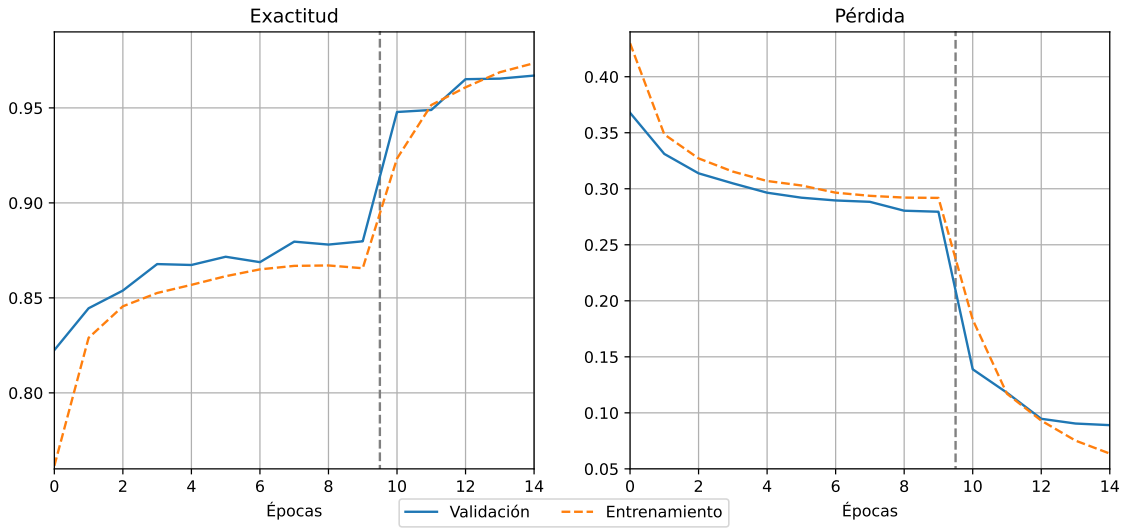
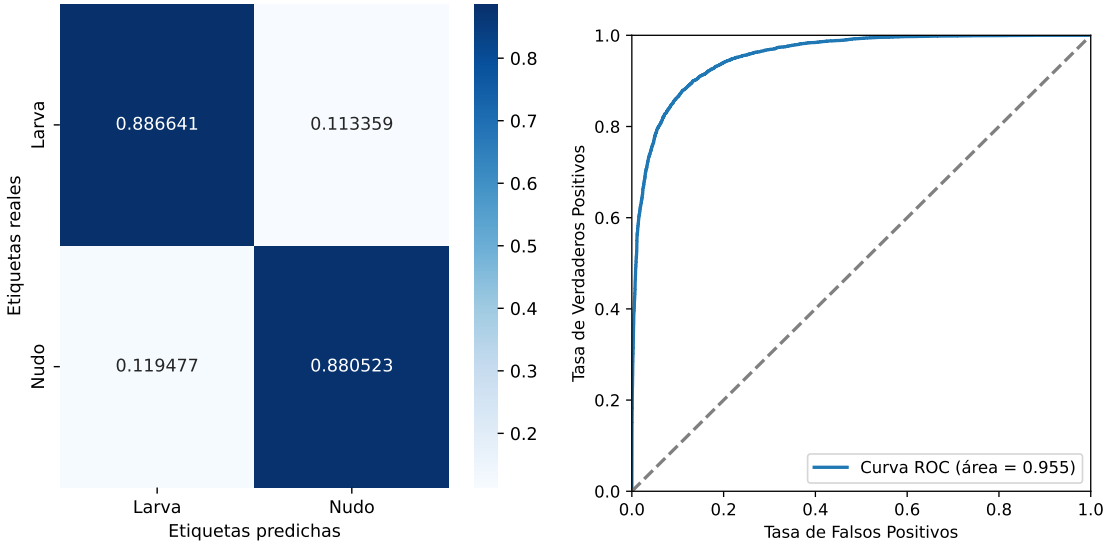


Figura 5.1: Curvas de aprendizaje del modelo basado en Xception.

Es importante el cambio que se produce cuando se "descongelan" los pesos del modelo base, puesto que previamente no se podían modificar gran parte de los valores aprendidos

por el modelo, y por ese motivo se había alcanzado un valor límite en la exactitud que no se conseguía mejorar. Es por eso que se observa la efectividad del *fine-tuning*, puesto que a partir de la época 10 la exactitud aumenta considerablemente, a la vez que cae la pérdida.



(a) Matriz de confusión.

(b) Curva ROC.

Figura 5.2: Evaluación del clasificador basado en Xception.

En cuanto a los resultados de clasificación realizados por el modelo, en la Figura 5.2 se detallan la matriz de confusión y la curva ROC sobre el conjunto de prueba, nunca antes "visto" por el modelo. A partir de la matriz de confusión se pueden calcular las siguientes métricas:

1. Exactitud: 0.884. Esto indica que el 88.4 % de los casos se ha clasificado correctamente, ya sean de larva o de nudo.

2. Precisión: 0.881. Esta métrica hace referencia a la exactitud de las predicciones positivas, es decir, el número de veces que se clasifica correctamente la señal de un nudo frente al total de predicciones de nudos, con lo que del 88.1 % de veces que se ha clasificado como nudo, realmente lo era.

3. Sensibilidad: 0.887. En este caso, la sensibilidad mide cómo de bien clasifica el modelo los nudos con respecto al número total de nudos reales, es decir, el 88.7 % de los nudos han sido clasificados correctamente.

4. Especificidad: 0.881. Este valor indica la proporción de larvas correctamente identificadas por el modelo, en este caso el 88.1 % de las señales correspondientes a una larva fueron correctamente clasificadas.

5. Puntuación F1: 0.884. La media armónica entre la precisión y la sensibilidad en este caso es del 88.4 %.

Por su parte, de la curva ROC se puede calcular el AUC, obteniendo un resultado de 0.955. En general, los valores de AUC mayores de 0.9 son considerados excelentes. [55]

5.2. CNN

Vista la eficacia del modelo basado en Xception, se esperan resultados similares con una CNN sencilla, diseñada y entrenada desde cero.

En la Figura 5.3 se muestran las curvas de exactitud y pérdida a lo largo del entrenamiento. En este caso, las curvas mostradas corresponden a un entrenamiento sin *dropout*, puesto que se alcanzaron valores más altos en términos de exactitud. A pesar de que el grado de estabilidad no es muy alto, como se puede observar en los saltos que va dando la curva de validación, este es el mejor resultado conseguido.

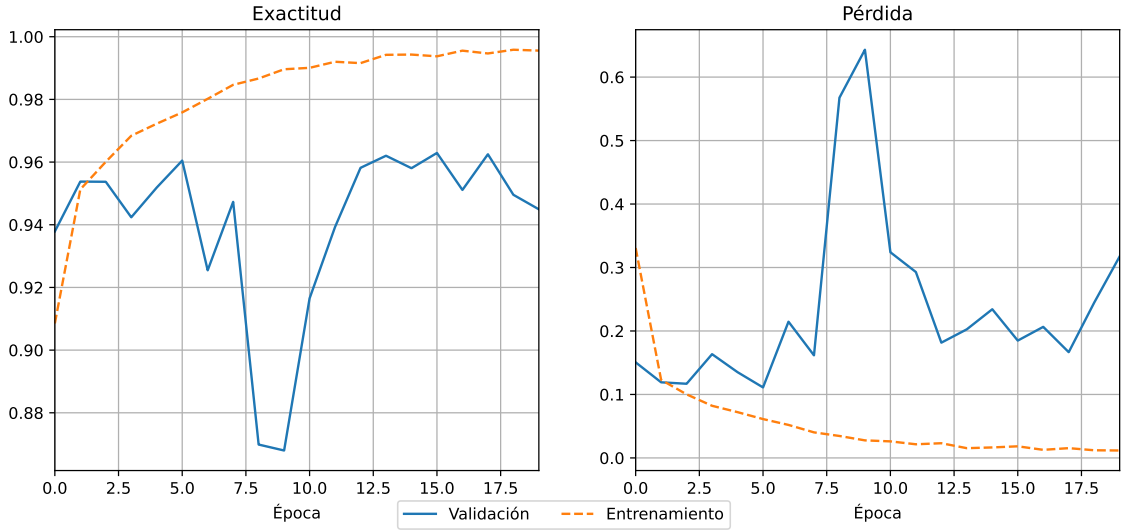


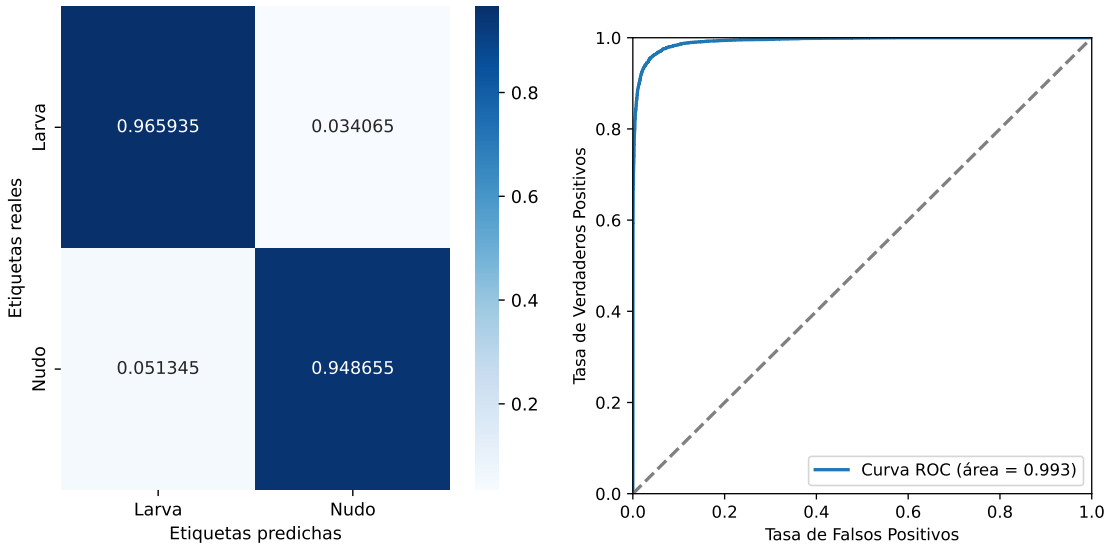
Figura 5.3: Curvas de aprendizaje del modelo de CNN.

Con unas pocas épocas se consigue una exactitud de 0.96, un valor superior al conseguido con Xception. Sin embargo, la curva de validación no se aproxima tanto a la de entrenamiento como en otros casos, sino que, aunque las curvas de entrenamiento mejoran, las de validación se mantienen igual o empeoran. Esto es típico de un problema de sobreajuste, puesto que a partir de la época 5 el modelo parece empeorar sobre los datos de validación.

Aún así, la evaluación del comportamiento del modelo resulta positiva. En la Figura 5.4 se observa que el poder de clasificación del modelo sobre el conjunto de prueba es excelente, y mejora en comparación con el anterior.

A partir de la matriz de confusión se pueden extraer las siguientes métricas:

1. Exactitud: 0.957. Esto indica que en el 95.7 % de los casos, se ha clasificado correctamente.
2. Precisión: 0.950. Esta métrica hace referencia a la exactitud de las predicciones positivas, es decir, el número de veces que se clasifica correctamente la señal de un nudo frente al total de predicciones de nudos, con lo que el 95 % de veces que se ha clasificado como nudo, lo era realmente.
3. Sensibilidad: 0.966. En este caso, la sensibilidad mide cómo de bien clasifica el modelo los nudos con respecto al número total de nudos reales, es decir, el 96.6 % de los nudos han sido clasificados correctamente.



(a) Matriz de confusión.

(b) Curva ROC.

Figura 5.4: Evaluación del modelo con arquitectura CNN como clasificador.

4. Especificidad: 0.949. Este valor indica la proporción de larvas correctamente identificadas por el modelo, en este caso el 94.9 % de las señales correspondientes a una larva fueron correctamente clasificadas.

5. Puntuación F1: 0.958. La media armónica entre la precisión y la sensibilidad en este caso es del 95.8 %.

De la curva ROC se puede calcular el AUC, obteniendo un resultado de 0.993, mejorando el resultado obtenido con el modelo Xception.

5.3. MLP

Una vez probados los modelos de redes convolucionales, a continuación se evaluará el comportamiento de los perceptrones multicapa para distintos datos de entrada.

5.3.1. Señal en el tiempo

En el caso del MLP entrenado con la secuencia temporal se obtuvieron las curvas de entrenamiento mostradas en la Figura 5.5.

Se puede apreciar un grado de estabilidad mucho mayor comparado con el resultante del entrenamiento de la CNN, y aunque a partir de la época 20 no mejoran demasiado las curvas de validación, sí que en general acompañan a las de entrenamiento. De la curva de pérdida se puede observar que entre la época 10 y 20 se alcanza el mínimo, a partir del cual la curva comienza a subir ligeramente, probablemente cayendo en una situación de sobreajuste.

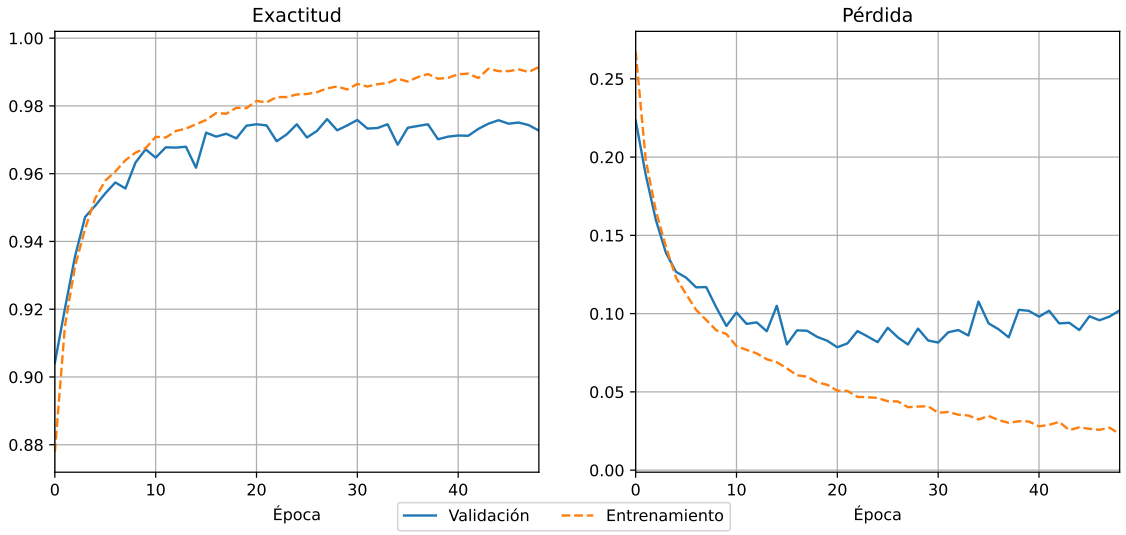
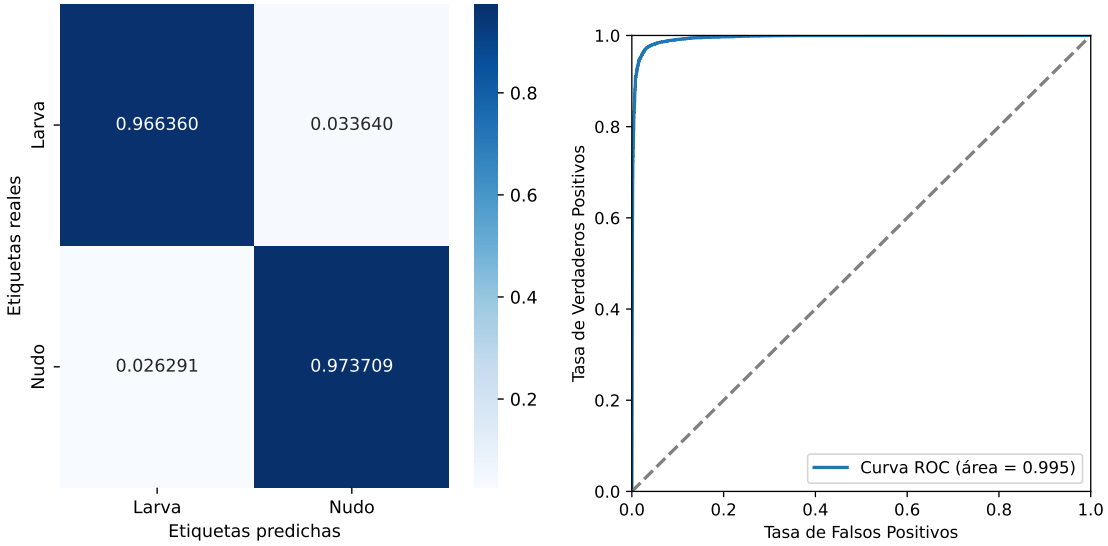


Figura 5.5: Curvas de aprendizaje del MLP usando la secuencia temporal.

En cuanto a los resultados de clasificación realizados por el modelo, se muestran en la Figura 5.6 la matriz de confusión y la curva ROC.



(a) Matriz de confusión.

(b) Curva ROC.

Figura 5.6: Evaluación del MLP usando la secuencia temporal como clasificador.

A partir de la matriz de confusión se pueden extraer las siguientes métricas:

1. Exactitud: 0.970. Esto indica que el 97 % de los casos se ha clasificado correctamente, ya sean de larva o de nudo.

2. Precisión: 0.974. Esta métrica hace referencia a la exactitud de las predicciones positivas, es decir, el número de veces que se clasifica correctamente la señal de un nudo frente al total de predicciones de nudos, con lo que del 97.4 % de veces que se ha clasificado como nudo, realmente lo era.

3. Sensibilidad: 0.966. En este caso, la sensibilidad mide cómo de bien clasifica el modelo los nudos con respecto al número total de nudos reales, es decir, el 96.6 % de los nudos han sido clasificados correctamente.

4. Especificidad: 0.974 Este valor indica la proporción de larvas correctamente identificadas por el modelo, en este caso el 97.4 % de las señales correspondientes a una larva fueron correctamente clasificadas.

5. Puntuación F1: 0.970. La media armónica entre la precisión y la sensibilidad en este caso es del 97 %.

Por su parte, la curva ROC proporciona una valor de AUC de 0.995, consiguiendo mejorar ligeramente el obtenido por la CNN.

5.3.2. Espectro en frecuencia

Además de entrenar el MLP con la secuencia temporal, se entrenó otro MLP con la DFT de dicha secuencia. Las curvas de aprendizaje de este nuevo modelo se presentan en la Figura 5.7.

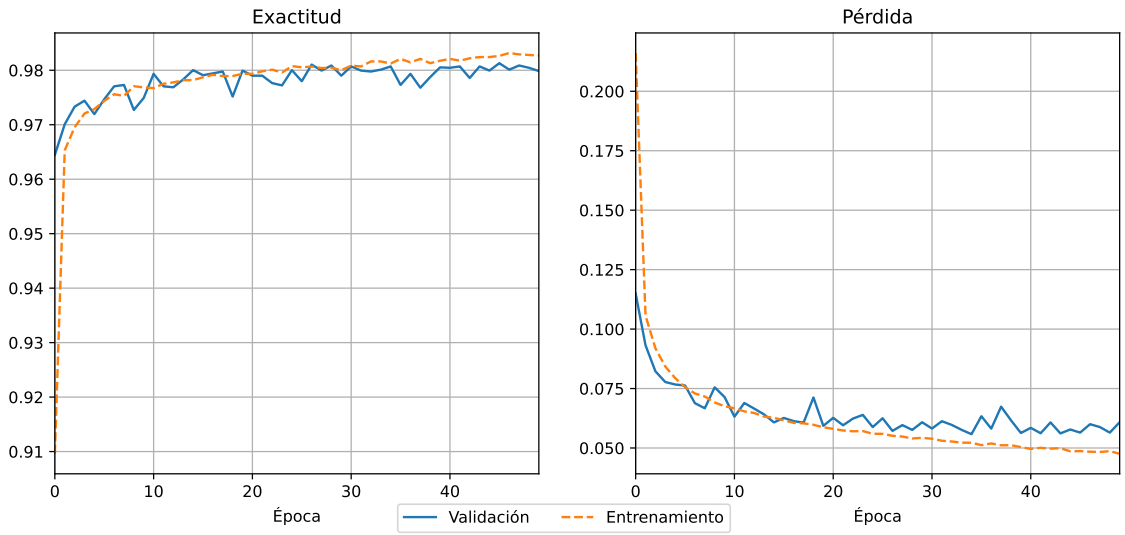
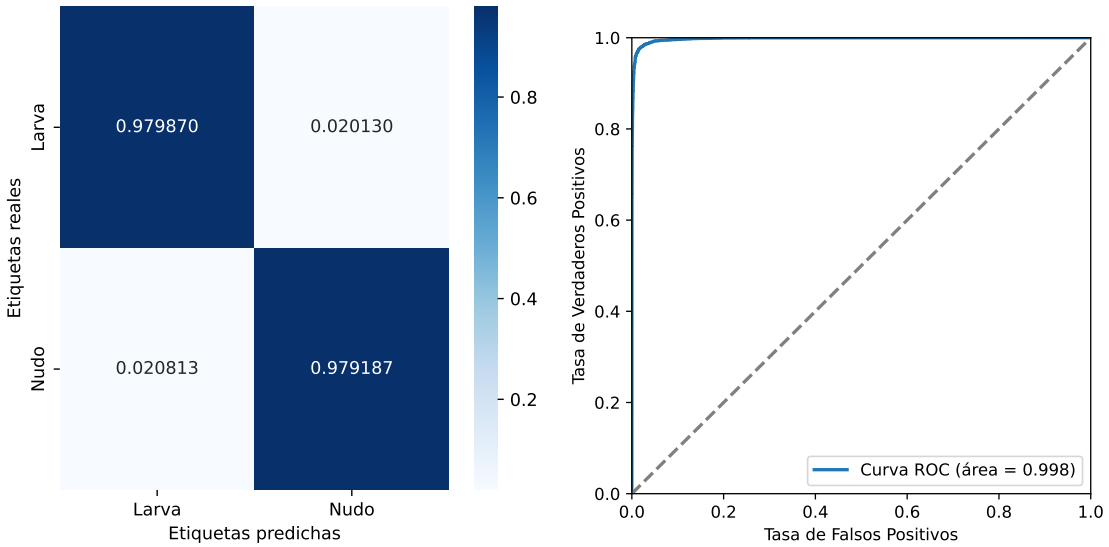


Figura 5.7: Curvas de aprendizaje del MLP usando la DFT.

Observando estas curvas, se puede apreciar que la curva de validación se mantiene muy cercana a la de entrenamiento, sin una gran distancia entre ambas. En ciertos puntos, la curva de validación incluso supera los resultados de la curva de entrenamiento, lo cual es ideal. Aunque alrededor de la época 50 las curvas comienzan a separarse un poco, no parece haber indicios de sobreajuste.

Para evaluar el poder de clasificación del modelo, en la Figura 5.8 se presentan la matriz de confusión y la curva ROC, ambas calculadas sobre el conjunto de prueba.

Con la matriz de confusión se han calculado las siguientes métricas para evaluar el comportamiento del modelo:



(a) Matriz de confusión.

(b) Curva ROC.

Figura 5.8: Evaluación del MLP usando la DFT como clasificador.

1. Exactitud: 0.980. Esto indica que el 98 % de los casos se ha clasificado correctamente, ya sean de larva o de nudo.
2. Precisión: 0.979. Esta métrica hace referencia a la exactitud de las predicciones positivas, es decir, el número de veces que se clasifica correctamente la señal de un nudo frente al total de predicciones de nudos, con lo que del 97.9 % de veces que se ha clasificado como nudo, realmente lo era.
3. Sensibilidad: 0.980. En este caso, la sensibilidad mide cómo de bien clasifica el modelo los nudos con respecto al número total de nudos reales, es decir, el 98 % de los nudos han sido clasificados correctamente.
4. Especificidad: 0.979. Este valor indica la proporción de larvas correctamente identificadas por el modelo, en este caso el 97.9 % de las señales correspondientes a una larva fueron correctamente clasificadas.
5. Puntuación F1: 0.980. La media armónica entre la precisión y la sensibilidad en este caso es del 98 %.

Por su parte, la curva ROC proporciona una valor de AUC de 0.998, consiguiendo un resultado ligeramente mejor que el alcanzado entrenando el MLP con la secuencia temporal.

En el caso del MLP entrenado con la DFT, se examinaron los pesos de las neuronas en la capa de entrada de la red, con el objetivo de identificar a cuáles de estas se les asignan mayores pesos en el proceso de toma de decisiones para la clasificación. Esos pesos se presentan en la Figura 5.9 como un diagrama de barras, donde cada neurona se representa mediante el índice de frecuencia correspondiente, y la altura de cada barra indica el peso asociado a esa neurona.

Si bien es cierto que todos los índices son importantes para la clasificación, claramente la red ha asignado valores mayores a las frecuencias más bajas, por lo que se trató de aplanar la respuesta frecuencial del array para conseguir un resultado mejor.

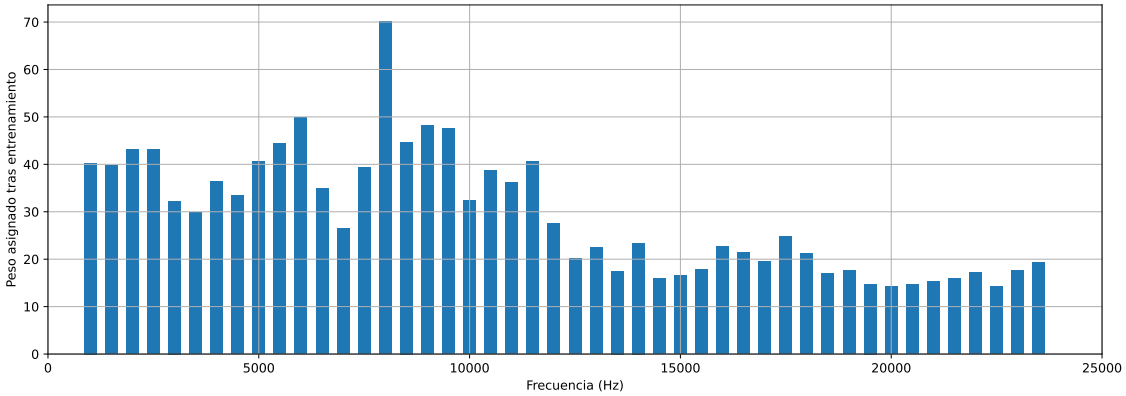


Figura 5.9: Pesos de las neuronas de la capa de entrada.

Con esta corrección de la respuesta se obtuvieron nuevos valores de DFT con los que se entrenó de nuevo el MLP. La curva de entrenamiento correspondiente se muestra en la Figura 5.10, donde se observa que se obtienen resultados similares, en algún punto algo más inestables.

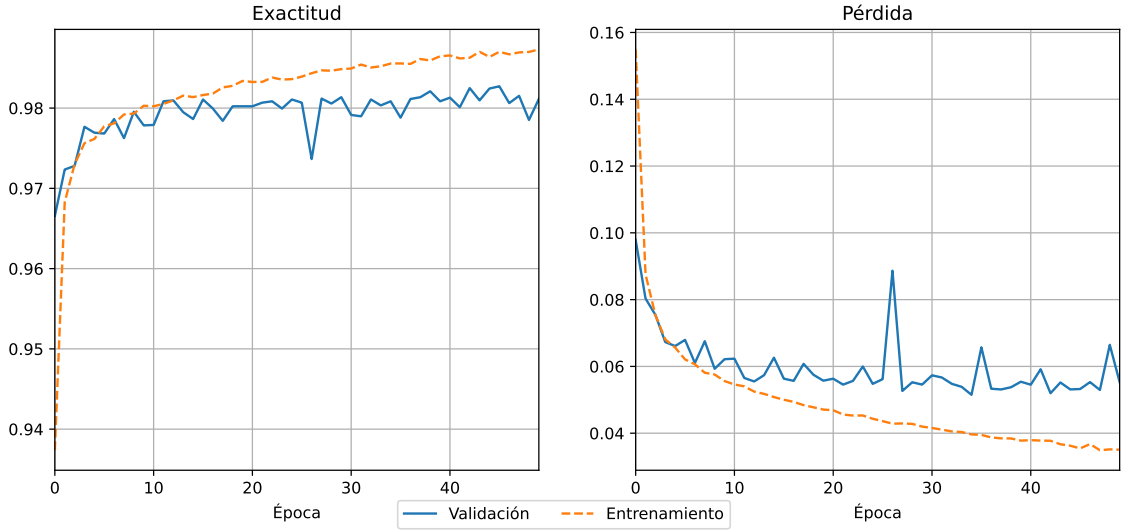


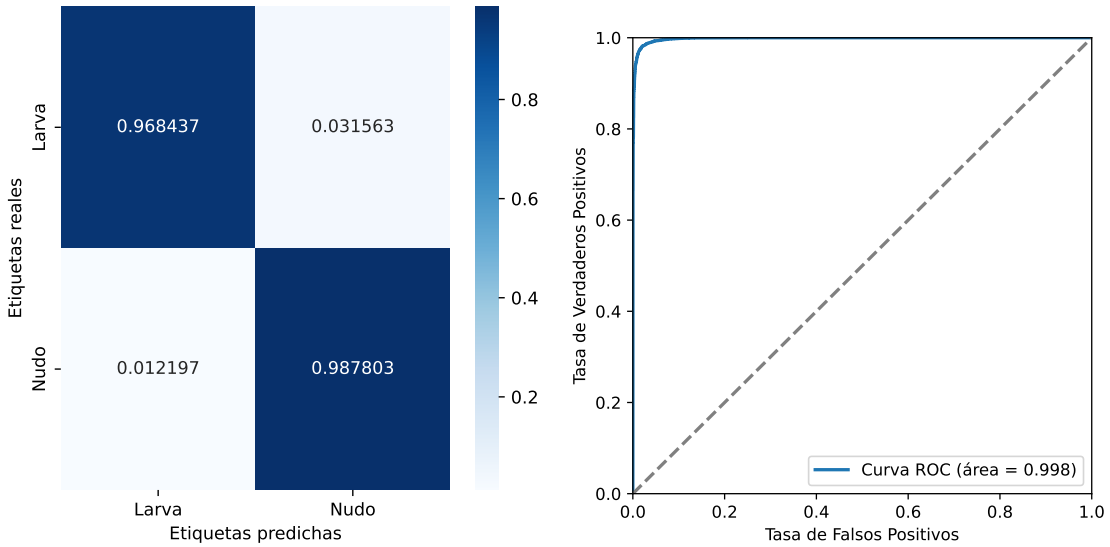
Figura 5.10: Curvas de aprendizaje del MLP usando la DFT tras la corrección de la respuesta frecuencial.

Con la matriz de confusión se han calculado las siguientes métricas para evaluar el comportamiento del modelo:

1. Exactitud: 0.978. Esto indica que el 97.8 % de los casos se ha clasificado correctamente, ya sean de larva o de nudo.

2. Precisión: 0.988. Esta métrica hace referencia a la exactitud de las predicciones positivas, es decir, el número de veces que se clasifica correctamente la señal de un nudo frente al total de predicciones de nudos, con lo que del 98.8 % de veces que se ha clasificado como nudo, realmente lo era.

3. Sensibilidad: 0.969. En este caso, la sensibilidad mide cómo de bien clasifica el modelo



(a) Matriz de confusión.

(b) Curva ROC.

Figura 5.11: Evaluación del MLP usando la DFT tras la corrección de la respuesta frecuencial como clasificador.

los nudos con respecto al número total de nudos reales, es decir, el 96.9 % de los nudos han sido clasificados correctamente.

4. Especificidad: 0.988. Este valor indica la proporción de larvas correctamente identificadas por el modelo, en este caso el 98.8 % de las señales correspondientes a una larva fueron correctamente clasificadas.

5. Puntuación F1: 0.979. La media armónica entre la precisión y la sensibilidad en este caso es del 97.9 %.

Por su parte, la curva ROC proporciona una valor de AUC de 0.998, un resultado excelente pero igual al alcanzado con el MLP con la DFT sin la normalización de la respuesta frecuencial, lo que involucra un paso de procesamiento adicional en este caso.

5.4. Validación *leave-one-out*

Para comprobar el poder de generalización del mejor modelo conseguido, aquel entrenado con la DFT, se realizó una validación *leave-one-out*. Esta técnica es una forma exhaustiva de validación cruzada donde se entrena el modelo tantas veces como sujetos haya en el conjunto de datos, dejando fuera a uno en cada iteración para evaluarlo.

El procedimiento realizado consistió en entrenar desde cero la red MLP utilizando las muestras de la DFT, ya que estas ofrecieron los mejores resultados. En cada iteración, se excluyó una de las larvas del conjunto de entrenamiento. Es decir, primero se entrenó un modelo que incluía las señales de todas las larvas excepto de la larva 1, y una vez entrenado, se utilizó ese modelo para hacer predicciones sobre las señales de la larva 1. Este proceso se repitió con el resto de larvas excluyendo en cada modelo a una de ellas, entrenando de esta manera seis modelos diferentes, cada uno diseñado para evaluar las

predicciones del modelo respecto a las señales de la larva que no había sido utilizada durante su entrenamiento.

En la Figura 5.12 se muestra un histograma normalizado con los valores predichos por cada modelo sobre las señales de la larva con la que no se ha entrenado. Recordemos que un `0` equivale a "Larva", mientras que un `1` representa al "Nudo". Los histogramas presentados muestran la frecuencia con la que se repiten los diferentes valores de salida de la última neurona del modelo, correspondientes a las distintas señales de la larva.

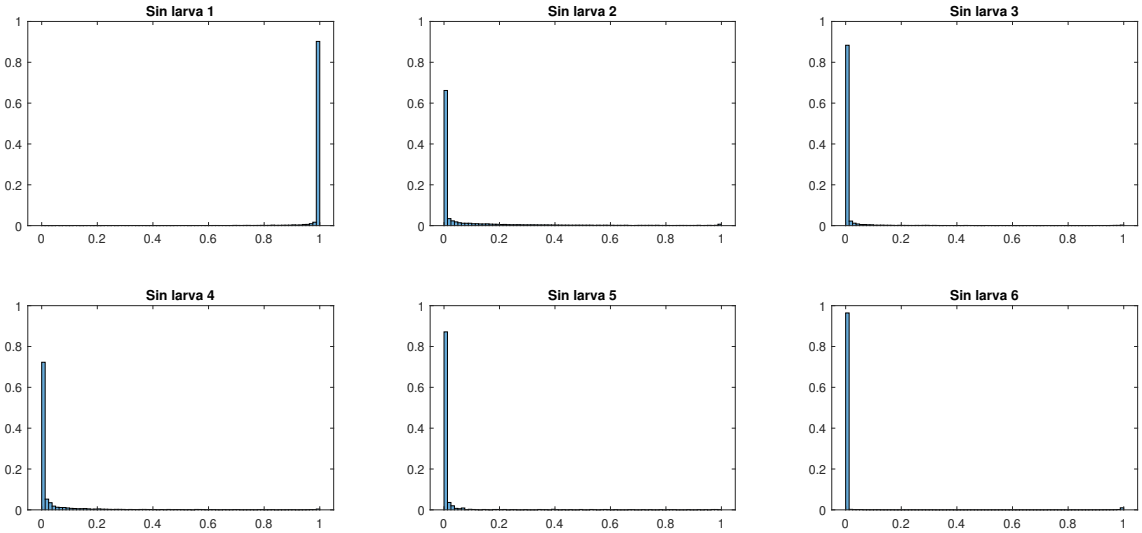


Figura 5.12: Histograma de las predicciones.

Por su parte, en la Figura 5.13 se establece un umbral de 0.5, de forma que cuando la salida del modelo sea superior a este valor, el resultado de la clasificación será "Nudo", mientras que si es menor, será "Larva". Por tanto todos los valores intermedios que se veían en los histogramas anteriores quedan agrupados en la categoría más próxima.

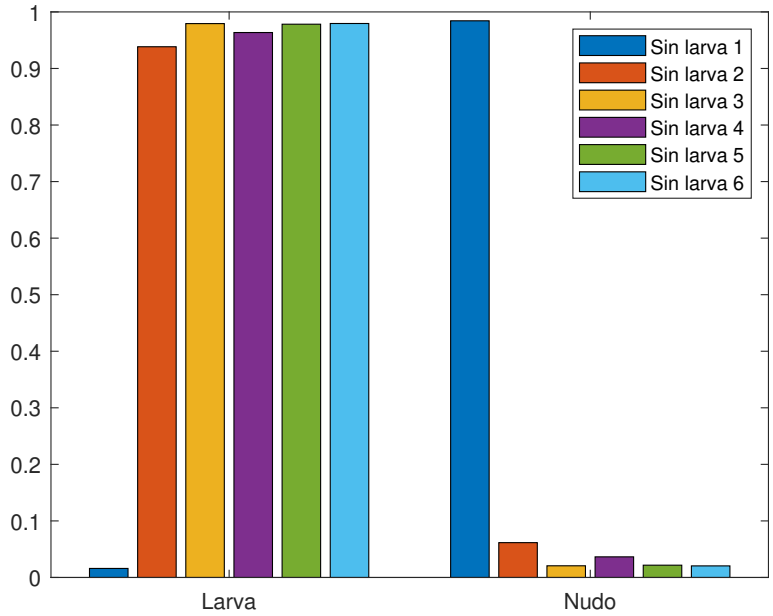


Figura 5.13: Resultado de predicciones con un umbral de 0.5.

Puesto que los datos de validación son siempre de larvas, idealmente todas las detecciones deberían clasificarse como tal, pero a partir de estos resultados, se observa que la mayoría de modelos tienen un comportamiento correcto en la mayoría de los casos, ya que cinco de los seis modelos identifican como larva todas las señales en más de un 93 % de los casos, lo que indica que los modelos están generalizando correctamente puesto que hacen predicciones correctas sobre datos nunca antes vistos. Sin embargo, el caso del modelo entrenado sin la larva 1 es problemático, puesto que clasifica más de un 98 % de las señales de esta larva como nudo. Esto se debe a que los sonidos del nudo 1 provienen esencialmente de la larva 1, pero al no tener los datos de la larva 1 durante el entrenamiento lo más parecido que se encuentra al hacer la predicción son los sonidos del nudo 1. Por este motivo es importante que el entrenamiento se lleve a cabo con los sonidos de las larvas que puedan estarse transmitiendo por nudos.

5.5. Comparativa

El criterio para elegir el mejor modelo para cada arquitectura fue escoger aquella época en la que el valor de pérdida fuese menor. En base a esto, se calcularon todas las métricas explicadas hasta ahora, cuyo resumen se expone en la Tabla 5.1.

	Xception	CNN	MLP $x(t)$	MLP $X(f)$	MLP $\overline{X}(f)$
Exactitud	0.884	0.957	0.970	**0.980**	0.978
Precisión	0.881	0.950	0.974	0.979	**0.988**
Sensibilidad	0.887	0.966	0.966	**0.980**	0.968
Especificidad	0.881	0.949	0.974	0.979	**0.988**
Puntuación F1	0.884	0.958	0.970	**0.980**	0.978

Tabla 5.1: Comparativa de las métricas calculadas sobre los distintos modelos.

En esta tabla se muestran el valor de cada métrica para cada modelo, donde $x(t)$ hace referencia a la secuencia temporal, $X(f)$ a la DFT de la señal, y $\overline{X}(f)$ a la DFT tras la normalización de la respuesta en frecuencia del array. En negrita se destaca el modelo con mejor resultado para cada métrica.

En cuanto al poder de discriminación de cada modelo, en la Figura 5.14 se muestran las curvas ROC obtenidas con cada modelo. Considerando estos resultados, se puede concluir que el modelo que presenta el mejor rendimiento es el MLP entrenado con la DFT, tanto con la normalización de la respuesta en frecuencia como sin ella. No obstante, el MLP entrenado con la secuencia temporal ofrece un valor de AUC muy similar y tiene la ventaja de no requerir cálculos adicionales, como es el caso del cálculo de la DFT. Por su parte, la CNN también muestra un buen rendimiento de clasificación, pero el coste computacional asociado al cálculo de los escalogramas es considerablemente mayor. Finalmente, el modelo basado en Xception preentrenado ayudó a demostrar que es posible discriminar si el sonido había atravesado un nudo o no, pero obtuvo el valor de AUC más bajo.

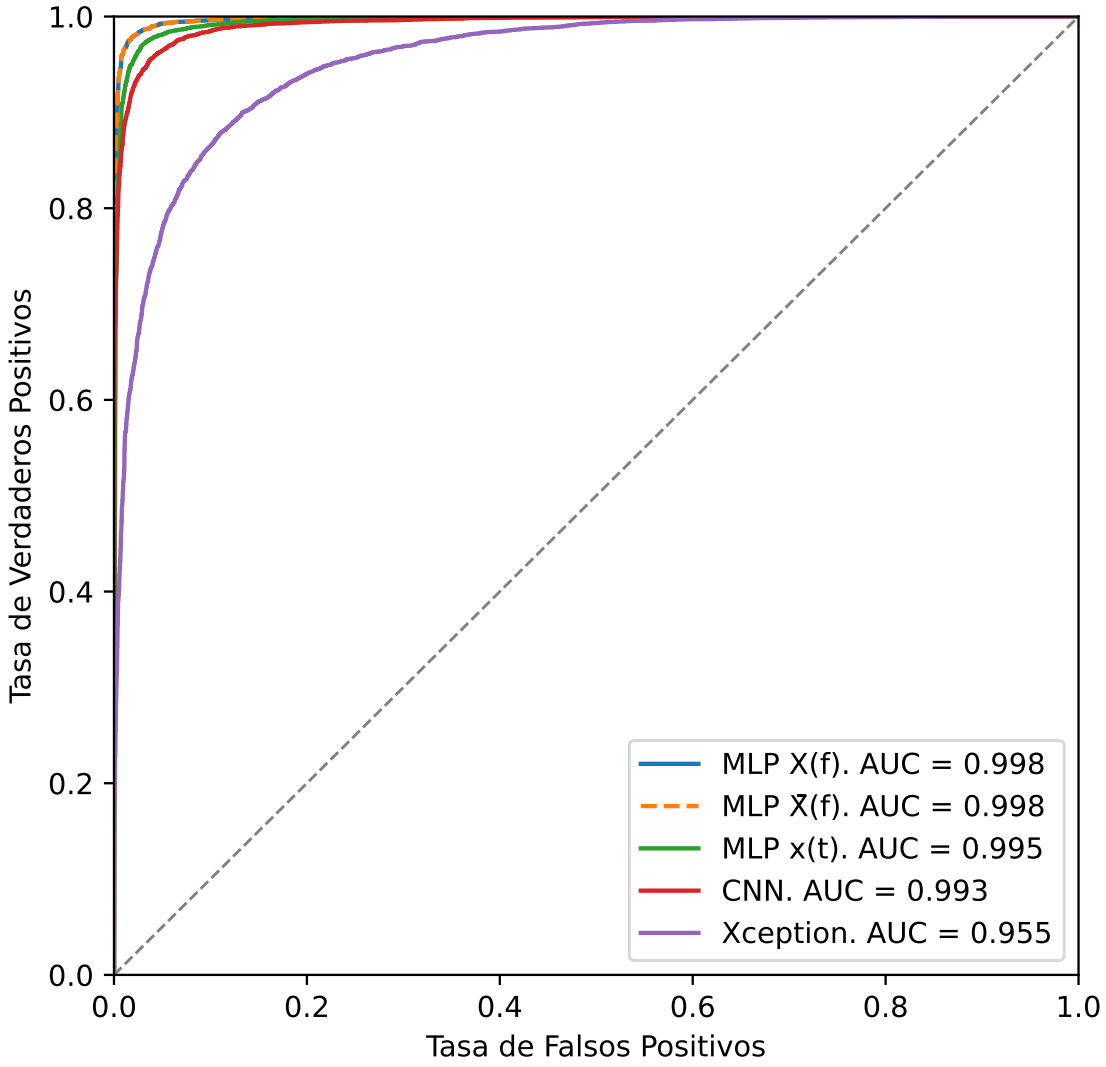


Figura 5.14: Comparativa de curvas ROC.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

Con este trabajo se ha conseguido mejorar la precisión en la detección de larvas de *Hylotrupes bajulus* gracias al aumento de datos conseguido a través de una segmentación multiblanco, que ha permitido multiplicar casi por cuatro el número de detecciones capturadas. Este aumento de datos ha facilitado también la detección de los dos nudos, ya que en [1] sólo se había detectado la presencia de uno de ellos, el nudo 1, puesto que ha sido el más activo. También ha sido una pieza clave para el entrenamiento de las redes neuronales, que precisan de una cantidad elevada de datos para su entrenamiento.

Además, el criterio de etiquetado empleado, basado en la frecuencia de detección de un evento acústico en cada coordenada de la cuadrícula, ha permitido acotar de una manera más efectiva la región a considerar para cada uno de los blancos. Esto ha facilitado la creación de un conjunto de datos bien etiquetado y representativo de la realidad, algo fundamental para el desarrollo de algoritmos de DL.

En cuanto al análisis de la evolución temporal realizado, este ha permitido encontrar relaciones entre los sonidos generados por cada uno de los blancos. Principalmente ha servido de ayuda para encontrar una fuerte correlación en la aparición de la larva 1 y su nudo asociado, el nudo 1, así como para observar las diferencias en la coexistencia de emisiones acústicas de la larva 2 y el nudo 2. La explicación a esto, es que la larva 1 se alimentó principalmente de madera que tenía más conexiones con el nudo 1, favoreciendo mucho la salida del sonido por este; mientras que la larva 2, se alimentó de madera con menos conexiones con el nudo 2, de manera que si mordía madera sin conexión con el nudo se detectaba su posición real, pero si se alimentaba de madera conectada con el nudo, la detección se localizaba en este. Además el tamaño del nudo 2 es significativamente menor.

Por su parte, las diferentes representaciones de las señales estudiadas han servido para determinar cuáles arrojan un mejor resultado en la posterior clasificación. A pesar de que la representación tiempo-frecuencia estudiada, a través de la Transformada Wavelet, ha permitido obtener buenos resultados de clasificación, se ha comprobado que la secuencia temporal y el espectro frecuencial calculado mediante la DFT han conseguido mejores resultados a pesar de requerir un coste computacional mucho menor. De la representación en frecuencia se ha extraído información interesante, mostrándose la importancia de las frecuencias más bajas para el resultado de clasificación.

En cuanto a los resultados de clasificación conseguidos, mostrados en el Capítulo 5, se puede concluir que la hipótesis de partida de que el sonido generado directamente por una larva, de aquel que atraviesa una fibra y es radiado por el nudo son significativamente distintos, aunque están fuertemente correlados.

El modelo de MLP entrenado con la DFT ha demostrado ser el más eficiente, ofreciendo un rendimiento superior en términos de precisión, exactitud y sensibilidad. Este modelo ha alcanzado una puntuación F1 de 0.979 y un AUC de 0.998, lo que indica una capacidad excepcional para diferenciar entre señales acústicas de insectos xilófagos y el sonido radiado a través de un nudo. La ventaja del MLP radica en su capacidad para equilibrar el rendimiento con un coste computacional relativamente bajo, lo que lo hace particularmente útil para aplicaciones prácticas donde los recursos computacionales pueden ser limitados, como sería un sistema portátil pensado para la localización de estos insectos en un escenario real.

Por otro lado, la CNN también ha mostrado un buen rendimiento con un AUC de 0.993, aunque su implementación implica un mayor coste computacional debido al cálculo intensivo de los escalogramas. Este hallazgo sugiere que, aunque las CNN son potentes, su aplicación puede ser más adecuada en entornos donde la capacidad computacional no sea un impedimento. Además, el modelo basado en Xception, aunque útil para discriminar si el sonido ha atravesado un nudo, ha obtenido un valor de AUC de 0.955, el más bajo entre los modelos evaluados, por lo que no es la mejor opción.

El estudio no solo ha evaluado la eficacia de los diferentes modelos de DL, sino que también ha subrayado su contribución a varios aspectos clave de la detección acústica de insectos xilófagos. En particular, el trabajo ha mostrado cómo las técnicas de aprendizaje profundo pueden mejorar significativamente la precisión de la detección, lo que es crucial para la implementación de métodos de control más efectivos y menos invasivos. La comparación detallada entre los modelos ha permitido identificar las fortalezas y debilidades de cada enfoque, proporcionando una base sólida para futuras investigaciones en el campo.

En conclusión, este trabajo ha demostrado que el uso de técnicas de DL en la detección acústica de insectos xilófagos no solo es viable sino también altamente efectivo. Los modelos desarrollados y evaluados proporcionan un marco robusto para la detección precisa y eficiente de estos insectos, lo que puede tener un impacto significativo en la gestión de plagas y la protección de estructuras de madera. La investigación realizada sienta las bases para futuras mejoras y adaptaciones de los modelos, asegurando su relevancia y efectividad en una variedad de contextos y aplicaciones.

6.2. Líneas futuras

Como ya se ha comentado anteriormente, uno de los principales problemas que presenta esta investigación es el número de individuos del que se extraen los datos. A pesar de que el número de señales capturadas es suficientemente alto para un problema de clasificación binaria usando un modelo de DL, la base de datos cuenta únicamente con 6 larvas y 2 nudos, por lo que es probable que el modelo pueda aprender bien a distinguir entre estas 6 larvas y estos 2 nudos, pero no generalice correctamente. Por tanto, aumentar el número de individuos bajo estudio sería muy conveniente.

Además de aumentar el número de individuos bajo estudio, podría ser útil que el sistema de

detección clasificara entre distintas especies de larva, de forma que el tratamiento necesario para la madera infestada sea el adecuado para el caso. Existen estudios previos que tratan este tema, Farr en su tesis de 2007 [19] consiguió diferenciar con un 96 % de precisión entre dos especies de escarabajos, *Hylotrupes bajulus* y *Prionus coriarius*, en base a su patrón de ocurrencia. Actualmente se está preparando una investigación para recoger señales de larvas de *Xestobium rufovillosum*, comúnmente conocido como el escarabajo del reloj de la muerte, lo cual permitirá establecer diferencias entre esta especie y la estudiada en este trabajo, *Hylotrupes bajulus*.

La especie de madera puede influir en los resultados de la clasificación. En este estudio, todas las vigas de madera eran de la especie *Pinus sylvestris L.*. Sin embargo, considerando que la fuente acústica es una combinación del sonido producido por la larva y la madera que la alberga, otras especies de madera, o incluso madera de la misma especie en diferentes estados, podrían afectar el sonido producido. Esto podría reducir la efectividad del clasificador.

En general, sería útil ampliar el conjunto de datos con más muestras de insectos y diferentes condiciones ambientales para mejorar la robustez del clasificador.

6.3. Objetivos de Desarrollo Sostenible

Los Objetivos de Desarrollo Sostenible (ODS) constituyen un llamamiento universal a la acción para poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo. En 2015, todos los Estados Miembros de las Naciones Unidas aprobaron 17 Objetivos, mostrados en la Figura 6.1, como parte de la Agenda 2030 para el Desarrollo Sostenible, en la cual se establece un plan para alcanzar los Objetivos en 15 años. [56]

Figura 6.1: Objetivos de Desarrollo Sostenible.

Este trabajo contribuye a algunos de estos Objetivos, sirviendo como un bien social en los siguientes aspectos:

1. ODS 9: Industria, innovación e infraestructura. Este trabajo fomenta la innovación tecnológica y la investigación científica aplicada al manejo de plagas, lo que puede mejorar la infraestructura relacionada con la protección de recursos naturales y la sostenibilidad.

2. ODS 11: Ciudades y comunidades sostenibles. Con métodos precisos de detección de insectos xilófagos se contribuye al mantenimiento preventivo de edificios históricos y de carácter patrimonial.

3. ODS 12: Producción y consumo responsables. La detección precisa de insectos xilófagos puede ayudar a manejar y reducir el uso de productos químicos en el control de plagas, promoviendo prácticas más sostenibles y responsables en la producción agrícola y la gestión forestal. Además, el uso de la detección acústica se podría aplicar en invernaderos.

4. ODS 15: Vida de ecosistemas terrestres. La detección y control efectivo de insectos xilófagos ayuda a proteger los ecosistemas terrestres, especialmente los bosques, que son cruciales para la biodiversidad y la salud del medio ambiente. Este trabajo contribuye a la lucha contra la desertificación y la degradación de las tierras, y ayuda a frenar la pérdida de biodiversidad. Además, se fomenta el uso de madera sostenible.

Bibliografía

[1] R. D. Martínez, A. Izquierdo, J. J. Villacorta, L. del Val, and L. A. Basterra, "Acoustic detection and localisation system for Hylotrupes bajulus L. larvae using a MEMS microphone array," *Applied Acoustics*, vol. 213, p. 109618, 10 2023. [Online]. Available: https://doi.org/10.1016/j.apacoust.2023.109618

[2] J. Schofield, "Real-time acoustic identification of invasive wood-boring beetles," 2011. [Online]. Available: https://etheses.whiterose.ac.uk/1978/

[3] G. Nicosia, H. Mathieu, J. L. Roux, A. de Wallens, and M. Dojat, "LarvaTracing: Imagerie RMN des infestations dans les œuvres d'art en bois et matériaux organiques," *In Situ*, 7 2019. [Online]. Available: https://doi.org/10.4000/insitu.22094

[4] J. Schofield and D. Chesmore, "Automated acoustic identification of beetle larvae in imported goods using time domain analysis," *Proceedings - European Conference on Noise Control*, pp. 5929–5934, 2008. [Online]. Available: https://www.researchgate.net/publication/5323731_Automated_acoustic_identification_of_beetle_larvae_in_imported_goods_using_time_domain_analysis

[5] A. Krajewski, S. Jakiela, and P. Witomski, "Detection of Old House Borer Larvae in Wooden Structures by Acoustic Emission Method – Influence of Larval Size and Sensor Location," *BioResources*, vol. 17, pp. 3435–3444, 2022. [Online]. Available: https://doi.org/10.15376/biores.17.2.3435-3444

[6] S. L. Conte, S. Vaiedelich, J. H. Thomas, V. Muliava, D. de Reyer, and E. Maurin, "Acoustic emission to detect xylophagous insects in wooden musical instrument," *Journal of Cultural Heritage*, vol. 16, pp. 338–343, 5 2015. [Online]. Available: https://doi.org/10.1016/j.culher.2014.07.001

[7] P. Bilski, P. Bobiński, A. Krajewski, and P. Witomski, "Detection of Wood Boring Insects' Larvae Based on the Acoustic Signal Analysis and the Artificial Intelligence Algorithm," *Archives of Acoustics*, vol. 42, pp. 61–70, 3 2017. [Online]. Available: https://doi.org/10.1515/aoa-2017-0007

[8] A. Krajewski, P. Bilski, P. Witomski, and P. Bobiński, "Assessment of the Ability for Early Detection of Newly Hatched Larvae of Hylotrupes bajulus L. Using the Acoustic Emission Method in Scots Pine Wood," *BioResources*, vol. 19, pp. 2092–2105, 5 2024. [Online]. Available: https://doi.org/10.15376/biores.19.2.2092-2105

[9] R. W. Mankin, A. Mizrach, A. Hetzroni, S. Levsky, Y. Nakache, and V. Soroker, "Temporal and Spectral features of Sounds of wood-boring Beetle Larvae: Identifiable patterns of Activity enable improved discrimination from background

noise," *Florida Entomologist*, vol. 91, p. 241 – 248, 2008. [Online]. Available: https://doi.org/10.1653/0015-4040(2008)91[241:TASFOS]2.0.CO;2

[10] R. W. Mankin, D. W. Hagstrum, M. T. Smith, A. L. Roda, and M. T. Kairo, "Perspective and Promise: a Century of Insect Acoustic Detection and Monitoring," *American Entomologist*, vol. 57, pp. 30–44, 1 2011. [Online]. Available: https://dx.doi.org/10.1093/ae/57.1.30

[11] R. Mankin, D. Hagstrum, M. Guo, P. Eliopoulos, and A. Njoroge, "Automated Applications of Acoustics for Stored Product Insect Detection, Monitoring, and Management," *Insects 2021, Vol. 12, Page 259*, vol. 12, p. 259, 3 2021. [Online]. Available: https://www.mdpi.com/2075-4450/12/3/259

[12] X. Liu, H. Zhang, Q. Jiang, L. Ren, Z. Chen, Y. Luo, and J. Li, "Acoustic Denoising Using Artificial Intelligence for Wood-Boring Pests Semanotus bifasciatus Larvae Early Monitoring," *Sensors 2022, Vol. 22, Page 3861*, vol. 22, p. 3861, 5 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/10/3861

[13] J. V. Oliver-Villanueva and M. A. Abián-Pérez, "Advanced wireless sensors for termite detection in wood constructions," *Wood Science and Technology*, vol. 47, pp. 269–280, 3 2013. [Online]. Available: https://doi.org/10.1007/S00226-012-0485-8

[14] A. Perles, R. Mercado, J. V. Capella, and J. J. Serrano, "Ultra-Low Power Optical Sensor for Xylophagous Insect Detection in Wood," *Sensors (Basel, Switzerland)*, vol. 16, 11 2016. [Online]. Available: https://doi.org/10.3390/S16111977

[15] H. L. V. Trees, *Optimum Array Processing*. Wiley, 3 2002.

[16] B. D. V. Veen and K. M. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE ASSP Magazine*, vol. 5, pp. 4–24, 1988.

[17] H. Unterwieser, G. Schickhofer, U. Di, and G. Schickhofer, "Influence of moisture content of wood on sound velocity and dynamic MOE of natural frequency- and ultrasonic runtime measurement," *European Journal of Wood and Wood Products*, vol. 69, pp. 171–181, 2 2010. [Online]. Available: https://hal.science/hal-00594483https://hal.science/hal-00594483/document

[18] I. Adachi, "Reproductive Biology of the White-Spotted Longicorn Beetle, Anoplophora malasiaca THOMSON (Coleoptera : Cerambycidae), in Citrus Trees," *Applied Entomology and Zoology*, vol. 23, pp. 256–264, 8 1988.

[19] I. Farr and E. Chesmore, "Automated bio-acoustic recognition of statutory qurantined insect pests: DFEFRA funded project PH0191," pp. 81–86, 2006. [Online]. Available: https://pure.york.ac.uk/portal/en/publications/automated-bio-acoustic-recognition-of-statutory-qurantined-insect

[20] F. Chollet, *Deep Learning with Python*, 1st ed. Manning, 2018.

[21] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*, 2nd ed. O'Reilly Media, Inc, 9 2019.

[22] J. Zou, Y. Han, and S. S. So, "Overview of artificial neural networks," *Methods in Molecular Biology*, vol. 458, pp. 15–23, 2008. [Online]. Available: https://link.springer.com/protocol/10.1007/978-1-60327-101-1_2

[23] A. Krogh, "What are artificial neural networks?" *Nature Biotechnology 2008 26:2*, vol. 26, pp. 195–197, 2 2008. [Online]. Available: https://www.nature.com/articles/nbt1386

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[25] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in Neural Networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 04, pp. 310–316, 5 2020. [Online]. Available: https://www.researchgate.net/publication/342195376_ACTIVATION_FUNCTIONS_IN_NEURAL_NETWORKS

[26] A. M. Yeves, "Sistema biométrico multimodal con imágenes acústicas y LiDAR utilizando una red neuronal," 2023. [Online]. Available: https://uvadoc.uva.es/handle/10324/63144

[27] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, pp. 2295–2329, 3 2017. [Online]. Available: https://arxiv.org/abs/1703.09039v2

[28] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature 2015 521:7553*, vol. 521, pp. 436–444, 5 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[29] K. L. Du, C. S. Leung, W. H. Mow, and M. N. Swamy, "Perceptron: Learning, Generalization, Model Selection, Fault Tolerance, and Role in the Deep Learning Era," *Mathematics 2022, Vol. 10, Page 4730*, vol. 10, p. 4730, 12 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/24/4730

[30] J. K. Yuen, E. W. Lee, S. M. Lo, and R. K. Yuen, "An intelligence-based optimization model of passenger flow in a transportation station," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1290–1300, 2013. [Online]. Available: https://www.researchgate.net/publication/258768635_An_Intelligence-Based_Optimization_Model_of_Passenger_Flow_in_a_Transportation_Station

[31] M. M. Taye, "Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions," *Computation 2023, Vol. 11, Page 52*, vol. 11, p. 52, 3 2023. [Online]. Available: https://www.mdpi.com/2079-3197/11/3/52

[32] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A Survey of Optimization Methods from a Machine Learning Perspective," *IEEE Transactions on Cybernetics*, vol. 50, pp. 3668–3681, 6 2019. [Online]. Available: https://arxiv.org/abs/1906.06821v2

[33] S. Ruder, "An overview of gradient descent optimization algorithms," 9 2016. [Online]. Available: https://arxiv.org/abs/1609.04747v2

[34] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature 1986 323:6088*, vol. 323, pp. 533–536, 1986. [Online]. Available: https://www.nature.com/articles/323533a0

[36] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, and E. A. Chavez-Urbiola, "Loss Functions and Metrics in Deep Learning," 7 2023. [Online]. Available: https://arxiv.org/abs/2307.02694v2

[37] J. A. M. Pérez and P. S. P. Martin, "ROC curve," *Semergen*, vol. 49, 1 2023. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/36155265/

[38] Z. H. Hoo, J. Candlish, and D. Teare, "What is an ROC curve?" *Emergency Medicine Journal*, vol. 34, pp. 357–359, 6 2017. [Online]. Available: https://emj.bmj.com/content/34/6/357https://emj.bmj.com/content/34/6/357.abstract

[39] J. R. L. Vizcaíno and J. P. Sebastiá, "LabVIEW: entorno gráfico de programación," p. 470, 2011. [Online]. Available: https://books.google.com/books/about/LabVIEW_Entorno_gr%C3%A1fico_de_programaci%C3%B3.html?hl=es&id=ZFQua3-eeQEC

[40] J. Kodosky and C. Lopes, "LabVIEW," *Proceedings of the ACM on Programming Languages*, vol. 4, 6 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3386328

[41] C. Moler, "A Brief History of MATLAB - MATLAB & Simulink," 2018. [Online]. Available: https://es.mathworks.com/company/technical-articles/a-brief-history-of-matlab.html

[42] P. J. G. and I. V. K., *Digital Signal Processing Using MATLAB*, 3rd ed. CL-EngineeringAmata Nakorn Industrial Estate, 1 2011.

[43] P. J. Costa, "MATLAB ® User's Guide Symbolic Math Toolbox Computation Programming Visualization Cleve Moler," 1993. [Online]. Available: http://www.mathworks.com

[44] S. Mihajlovic, A. Kupusinac, D. Ivetic, and I. Berković, "The Use of Python in the field of Artifical Intelligence," 10 2020. [Online]. Available: https://www.researchgate.net/publication/366578422_The_Use_of_Python_in_the_field_of_Artifical_Intelligence

[45] P. Naik, "Conceptualizing Python in Google COLAB," 2022. [Online]. Available: https://www.researchgate.net/publication/357929808

[46] Knowles, "SPH0641LU4H-1. Digital Zero-Height SiSonic Microphone With Multi-Mode And Ultrasonic Support." [Online]. Available: https://www.knowles.com/docs/default-source/model-downloads/sph0641lu4h-1-revb.pdf

[47] N. Instruments, "sbRIO-9607 - NI." [Online]. Available: https://www.ni.com/es-es/shop/model/sbrio-9607.html

[48] Mathworks, "Morse Wavelets." [Online]. Available: https://es.mathworks.com/help/wavelet/ug/morse-wavelets.html

[49] D. Zhang, *Wavelet Transform*, 2019, pp. 35–44. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-17989-2_3

[50] A. V. Oppenheim and R. W. Schafer, *Tratamiento de señales en tiempo discreto*, 3rd ed. Pearson Educacion, S.A., 2012, pp. 610–699.

[51] Y. Gong, Y. A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 1, pp. 56–60, 4 2021. [Online]. Available: https://arxiv.org/abs/2104.01778v3

[52] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, 10 2016. [Online]. Available: https://arxiv.org/abs/1610.02357v3

[53] ——, "Transfer learning & fine-tuning," 2020. [Online]. Available: https://keras.io/guides/transfer_learning/

[54] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition*, 2nd ed. Heaton Research, Inc., 2008.

[55] T. C. F. Polo and H. A. Miot, "Use of ROC curves in clinical and experimental studies," *Jornal Vascular Brasileiro*, vol. 19, pp. 1–4, 2020. [Online]. Available: /pmc/articles/PMC8218006/https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8218006/

[56] ONU, "La Agenda para el Desarrollo Sostenible - Desarrollo Sostenible." [Online]. Available: https://www.un.org/sustainabledevelopment/es/development-agenda/