Detección de sirenas para conductores con discapacidad auditiva con CNN

Archibald Emmanuel Carrion Claeys*, Fabián Vega Meza†, Marlon Esteban Murillo Quesada‡ y Edgar Casasola Murillo§
Escuela de Computación, Universidad de Costa Rica
San Jose, Costa Rica
Email: *archibald.carrion@ucr.ac.cr, †fabian.vegameza@ucr.ac.cr, ‡marlon.murilloquesada@ucr.ac.cr, §edgar.casasola@ucr.ac.cr

***Resumen*—Este proyecto presenta el desarrollo de un sistema basado en redes neuronales convolucionales (CNN) para la detección de sirenas de ambulancia en entornos urbanos de San Jose, Costa Rica. Utilizando un enfoque que combina la recolección de datos en campo y fuentes de internet, se transforman los audios en espectrogramas para identificar los sonidos de sirenas. Se proponen tres arquitecturas de CNN para abordar este problema y se realiza una comparación entre ellas para destacar sus distintas características y desempeños en la detección de los sonidos de sirenas proporcionados.**

***Abstract*—This project presents the development of a system based on Convolutional Neural Networks (CNNs) for detecting ambulance sirens in urban environments in San Jose, Costa Rica. Using an approach that combines data collection in the field and from internet sources, audio is transformed into spectrograms to identify siren sounds. Three CNN architectures are proposed to address this problem, and a comparison is conducted among them to highlight their different characteristics and performances in detecting the provided siren sounds.**

I. Justificación

La detección de sirenas de ambulancia en entornos urbanos es crucial para la seguridad y la movilidad en las vías públicas, siendo aún más desafiante para las personas con discapacidad auditiva, quienes no pueden percibir estos sonidos a distancia sin visualizar las luces de emergencia. Este proyecto tiene como objetivo intentar abordar esta limitación mediante el desarrollo de una red neuronal convolucional capaz de identificar sonidos de ambulancia dado un audio, con intención de que futuros proyectos puedan implementarlo a un sistema físico accesible capaz de identificar y alertar visualmente sobre la presencia de ambulancias cercanas. En Costa Rica, donde se permite a las personas con discapacidad auditiva conducir, esta tecnología adquiere una relevancia adicional al proporcionar una herramienta esencial para mejorar su autonomía y seguridad durante situaciones de emergencia en las vías públicas.

II. Objetivo general

El objetivo general de esta investigación es desarrollar e implementar un sistema basado en inteligencia artificial capaz de detectar las sirenas de ambulancias costarricenses a partir de grabaciones de audios tomados en ambientes del área de San Jose. Además diseñar y crear un conjunto de datos para llevar a cabo la experimentación.

III. Objetivos específicos

- Diseñar y crear un conjunto de datos conformado de sonidos grabados en Costa Rica para optimizar el rendimiento de nuestro sistema.
- Desarrollar tres arquitecturas de redes neuronales convolucionales para clasificar audios entre los que contienen sonido de sirenas de ambulancia y los que no.
- Evaluar el desempeño de tres arquitecturas diferentes planteadas de redes neuronales convolucionales
- Analizar posibles soluciones de mejora.

IV. Antecedentes

La detección de sirenas en el tráfico urbano ha sido objeto de estudio de varias publicaciones debido a su impacto en la seguridad vial y la eficiencia en la respuesta de los servicios de emergencia. A continuación, se presenta un resumen de algunos avances importantes en esta área.

- **[1] R. A. Dobre et al., "Low Computational Method for Siren Detection,"2015.** Un sistema de asistencia a la conducción que pueda detectar el sonido de sirenas y advertir al conductor. Se propuso una solución de bajo costo, en la que los sonidos de sirenas son interpretados por un Filtro paso banda de múltiples entradas para detectar las frecuencias comunes de una sirena de ambulancia en Rumanía. Después se pasó por diferentes componentes analógicos para determinar si el sonido es de ambulancia. El sistema fue simulado utilizando MATLAB y SPICE, mostrando resultados prometedores en la detección eficiente de sirenas.
- **[2] D. Pramanik et al., "Deep learning based urban sound classification and ambulance siren detector using spectrogram"2021.** Este artículo describe el desarrollo de una arquitectura para la detección de sirenas de ambulancia y la clasificación de sonidos urbanos utilizando diversas transformaciones de sonido a imagen, como Mel-spectrogramas, Scalogramas y el método de descomposición de Fourier (FDM). Esta investigación analiza los efectos de las técnicas de aumento y preprocesamiento en la eficacia de la arquitectura desarrollada frente a modelos preentrenados. El rendimiento del algoritmo propuesto se probó en el conjunto de datos Urbansound8K para

la clasificación de sonidos urbanos y en un conjunto de datos de múltiples fuentes para la detección de sirenas de ambulancia. La CNN propuesta alcanzó una precisión del 89.66 en la clasificación de sonidos urbanos y del 99.35 en la detección de sirenas

- **[3] Fatimah et al., .^n Automatic Siren Detection Algorithm Using Fourier Decomposition Method and MFCC"2020.** Se propone un algoritmo automático para la detección de sirenas utilizando la descomposición de Fourier y los coeficientes de cepstrum de frecuencia mel (MFCC). Este sistema emplea algoritmos de aprendizaje supervisado, como SVM(Máquinas de vectores de soporte) y árboles bagged ensemble, para entrenar modelos que distinguen las sirenas de otros ruidos de tráfico en la India.

V. Marco Teórico

A. Descripción de una CNN

Las CNN [4] son esencialmente una red neuronal que emplea la operación de convolución (en lugar de una capa completamente conectada) como una de sus capas, lo cual permite aprender características espaciales y temporales dentro de los datos. Esta estructura se inspira en la capacidad del sistema nervioso biológico para detectar y procesar patrones en imágenes de manera eficiente y robusta.

B. Elementos de una CNN

Las CNN están compuestas por tres capas principales: la capa de Convolución, la capa de pooling o agrupación, y la capa totalmente conectada.

- **Capa de Convolución:** La capa de convolución se utiliza para extraer características locales relevantes de las imágenes, como bordes, texturas y patrones de color. Esta emplea filtros o kernels que exploran la imagen y capturan patrones específicos mediante operaciones de multiplicación y suma de elementos, similar a detectar cambios rápidos de intensidad de color en un área específica de la imagen.
- **Capa de Pooling:** La capa de pooling tiene como objetivo reducir la dimensionalidad de la representación de la imagen mientras conserva las características más importantes para el análisis y reconocimiento de patrones. La imagen se divide en regiones y se aplica una operación de agregación como el máximo o el promedio sobre cada región, preservando características críticas como formas y texturas sin perder detalles importantes. Esta capa reduce la cantidad de datos a procesar, lo que disminuye el costo computacional y la complejidad del modelo.
- **Capa totalmente conectada:** En la capa totalmente conectada, todas las características aprendidas por las capas de convolución y pooling se utilizan para realizar la clasificación o regresión final de la imagen. Cada unidad neuronal en esta capa está conectada a todas las unidades de la capa anterior (generalmente la última capa de pooling), permitiendo que cada neurona considere todas las características extraídas en las etapas anteriores para hacer predicciones finales sobre la imagen.

C. Tensorflow

[5] Tensorflow es una plataforma creada por Google para desarrollar proyectos de aprendizaje automático, utilizando una serie de herramientas para entrenar y ejecutar modelos de redes neuronales. Esta biblioteca contiene una amplia variedad de herramientas para implementar redes convolucionales. Cuenta con funcionalidaddes usadasen el proyecto como, capas convolucionales, de pooling y totalmente conectada. Las capas convolucionales se implementan mediante la clase tf.keras.layers.Conv2D, que es parte del módulo Keras de TensorFlow. Esta clase permite definir y configurar capas convolucionales de manera sencilla. Se especifica el número de filtros (o kernels) y el tamaño del kernel. Además de la función de activación aplicada en ella.

- **Función de activación:** Estas funciones transforman la salida de una neurona, permitiendo que el modelo capture patrones no lineales. Tres de las funciones de activación más utilizadas son Sigmoid, ReLU (Rectified Linear Unit) y tanh.
- **Rectified Linear Unit(ReLU):** Esta función genera una salida igual a cero cuando la entrada sea negativa, y una salida igual a la entrada cuando esta sea positiva. Además, no cuenta con saturación, lo que causa que converja más rápidamente y sea computacionalmente poco costosa.
- **Tangente Hiperbólica (tanh):** La función tanh produce valores en el intervalo de -1 a +1. Esta si cuenta con mayores problemas de saturacion que relu, pero tiene mejor rendimiento con entradas negativas que ReLu.
- **Función de Activación Sigmoid :** Los valores de salida de la función sigmoid están en el intervalo de 0 a 1. Esto convierte cualquier entrada en una probabilidad, lo que ayuda en la clasificación binaria. Esta función es continua, lo que facilita la propagación de los gradientes durante el entrenamiento.

D. Espectrograma

Un espectrograma [6] es una herramienta básica de representación que se utiliza para el análisis de una señal eléctrica, de comunicación, o audiovisual, en la que se visualiza la energía del contenido frecuencial de la señal, según va variando a lo largo del tiempo. En otras palabras, es una representación visual o una conversión a imagen de la intensidad de las frecuencias de una señal de audio a lo largo del tiempo.

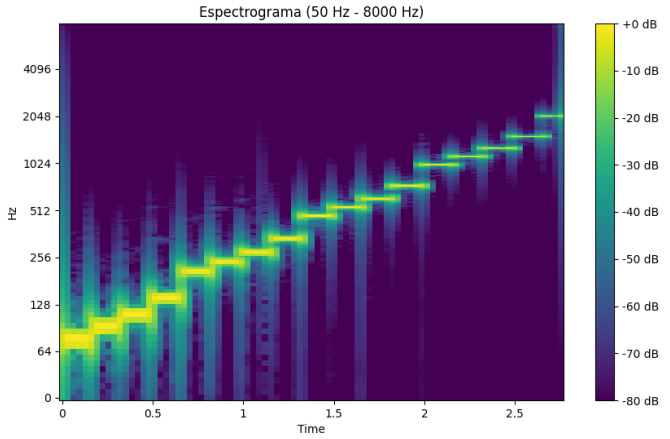

Figura 1. Ejemplo de un espectrograma..

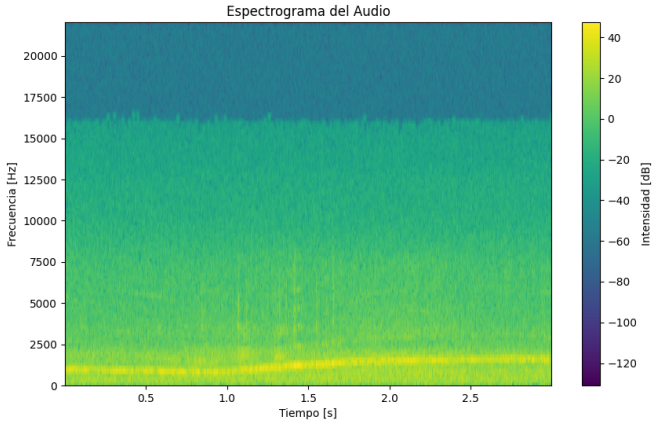

Figura 2. Espectrograma de un sonido de sirena.

La representación se realiza en tres dimensiones: temporal (en el eje x), frecuencial (en el eje y) y amplitud (representada por el color). En la figura 1 se muestra un ejemplo de espectrograma que representa gráficamente una escala musical ascendente.

VI. Metodología

A. Conjunto de datos para el entrenamiento

Para llevar a cabo el proyecto, primero se creó un conjunto de datos para el entrenamiento de la red neuronal. Se obtuvieron audios a partir de varias grabaciones de campo que se hicieron en las vías públicas del área de San Jose, además de varios videos de YouTube que contenían sirenas de ambulancia y sonidos ambientales de zonas urbanas costarricenses. [7] Se recolectaron muchos mas audios sin sirenas, por lo que se aplico la tecnca de SMOTE en la cual se hace un sobremuestreo de minorías sintéticas para aumentar el número de casos de un conjunto de datos de forma equilibrada. Estos audios fueron segmentados en fragmentos de 3 segundos en formato .wav (waveform audio format). Los fragmentos fueron clasificados en dos categorías: sonidos de ambulancia y sonidos ambientales sin sirenas. Al momento de entrenar y probar los modelos, se usaron 2 subdivisiones aleatorias del conjunto de datos: una primera subdivisión para el entrenamiento del modelo, y una segunda para la prueba del modelo.

B. Preprocesamiento de los datos

Los datos se pasaron por un proceso de estandarización para poderlos ingresar a las CNN de manera correcta. A estos audios se les aplicó una operación de preprocesamiento en la cual los audios se normalizaron a una frecuencia de 16 kHz y se convirtieron a formato mono de un solo canal. Posteriormente, los audios de entrenamiento procesados fueron convertidos en espectrogramas.

C. Entrenamiento del Modelo

En la figura 2, se muestra el espectograma de uno de los audios utilizados para el entrenamiento. La línea amarilla que aparece en la parte inferior del espectrograma representa la sirena de una ambulancia. Podemos observar que es la señal con mayor amplitud, lo que se evidencia por su color amarillo intenso.

Después de realizar todos los preprocesamientos en los datos, se procede a entrenar las redes neuronales convolucionales utilizando el conjunto de datos preparado. Se exploraron tres arquitecturas con diferentes parámetros en sus capas, incluyendo variaciones en las funciones de activación como ReLU y tanh, además de la adición de capas de pooling en la red.

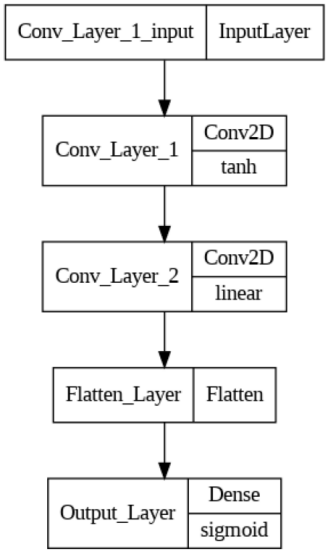

Figura 3. Arquitectura de la primera red

La primera arquitectura (ver figura 3) presenta una capa de entrada, dos convolucionales con función de activacion tanh, seguido de una capa de aplanamiento (o flatten layer en inglés)

que se usa para convertir una matriz multidimensional en un vector unidimensional. Dentro de las capas convolucionales, se utilizan 32 filtros en la primera y 64 en la segunda. Este proceso es esencial cuando se pasa de las capas convolucionales a las capas densas, tal como es nuestra capa de salida.

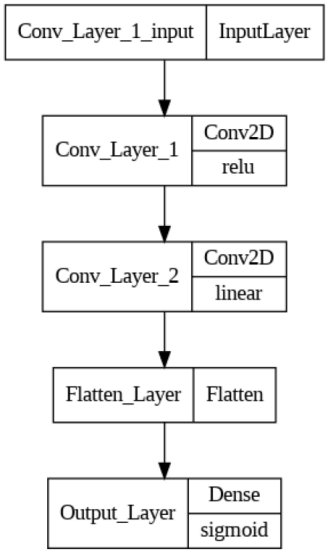

Figura 4. Arquitectura de la segunda red

La segunda arquitectura (ver figura 4) sigue un formato bastante similar al primero ya que tambien presenta una capa de entrada, dos convolucionales, seguido de una capa de aplanamiento que se usa para convertir una matriz multidimensional en un vector unidimensional. La mayor diferencia es que las capas convolucionales usan la funcion de activacion relu, la cual tiende a ser menos compleja computacionalmente hablando.

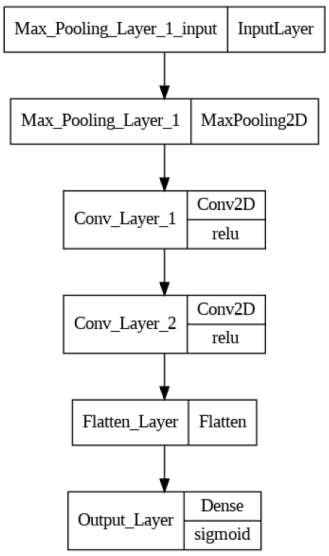

Figura 5. Arquitectura de la tercera red

La tercera arquitectura (ver figura 5) sigue un patrón similar a la arquitectura analizada previamente, con la diferencia de que previo a las capas de convolución se hace una operación en la cual se extrae la información más representativa: la matriz. Esto se hace por medio de la capa de MaxPooling, la cual selecciona los valores más altos de una entrada, para crear una segunda más reducida.

VII. Resultados

Todos los resultados presentados en el cuadro I. Fueron obtenidos con 5 epocas.

Cuadro I
Tiempos de ejecución de cada arquitectura.

	Tiempos de entrenamiento	Tiempos de prueba
Arquitectura 1	812,339	27,761
Arquitectura 2	771,902	27,827
Arquitectura 3	191,327	21,768

A. Arquitectura 1

Al entrenar la primera arquitectura propuesta, se nota que presenta una matriz de confusión casi perfecta, con únicamente dos falsos positivos. Se puede apreciar lo anterior viendo la matriz de la figura 6 .

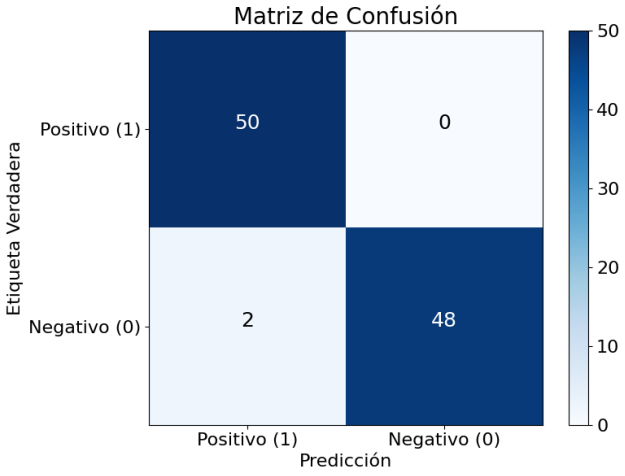

Figura 6. Matriz de Confusión generada tras probar la primera Arquitectura.

Es importante notar que el entrenamiento (ver cuadro I) dura 812 segundos, además para las pruebas dura un total de 27.7 segundos. Por lo que, por complejidad temporal, es el tercero mejor.

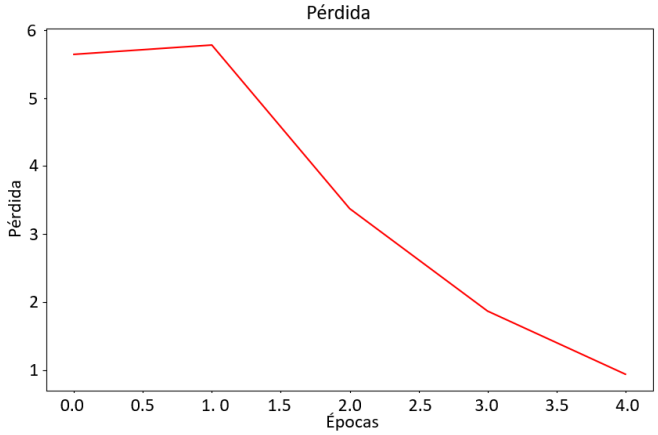

Figura 7. Grafo de pérdida generado durante el entrenamiento de nuestra primera Arquitectura.

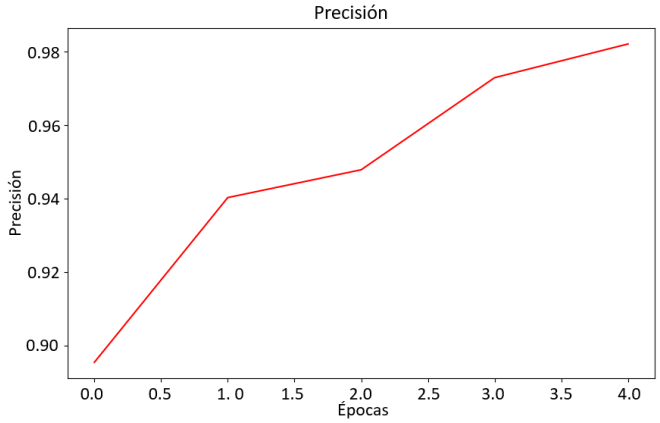

Figura 8. Grafo de precisión generado durante el entrenamiento de nuestra primera Arquitectura.

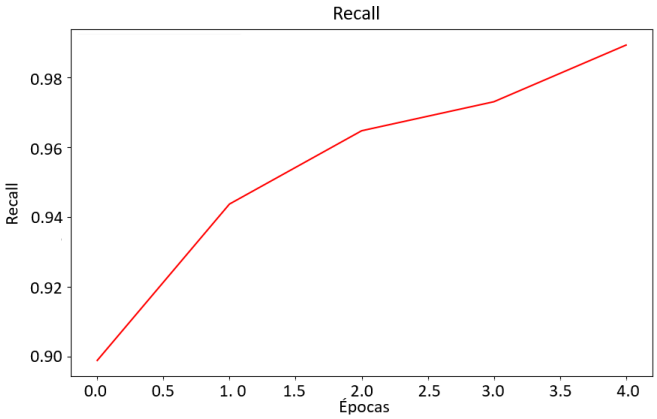

Figura 9. Grafo de recall generado durante el entranamiento de nuestra primera Arquitectura.

B. Arquitectura 2

Al entrenar la segunda arquitectura propuesta, se puede notar que también se obtuvo una matriz de confusión casi perfecta (ver figura 10) , que al igual que la primera arquitectura, solo optuvo dos falsos positivos.

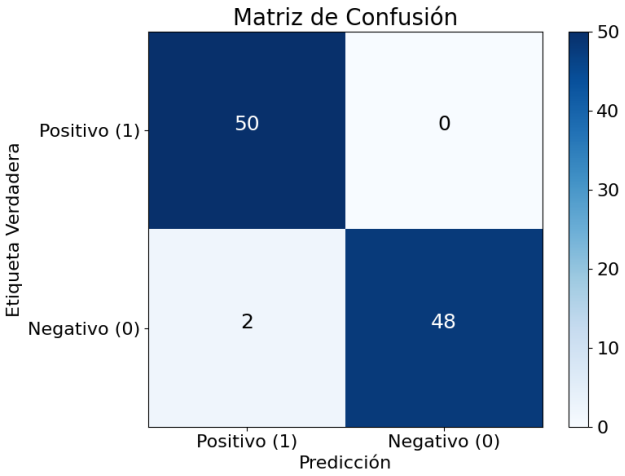

Figura 10. Matriz de Confusión generada tras probar la segunda Arquitectura.

Para la arquitectura 2 (ver cuadro I) el entrenamiento dura 771 segundos, y para las pruebas dura un total de 27.8 segundos. Por lo que por complejidad temporal es el segundo mejor, teniendo un rendimiento muy similar al anterior.

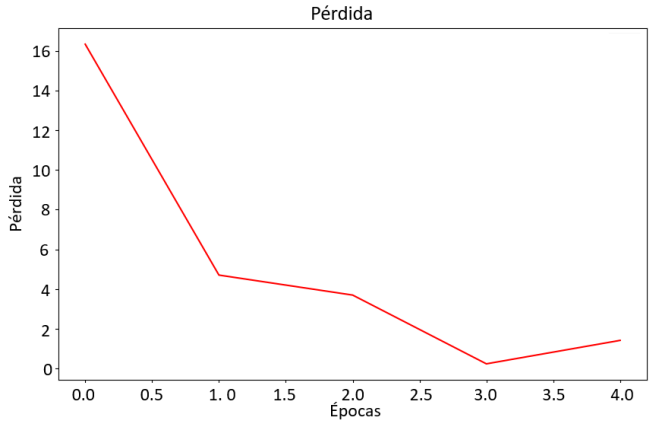

Figura 11. Grafo de pérdida generado durante el entrenamiento de nuestra segunda Arquitectura.

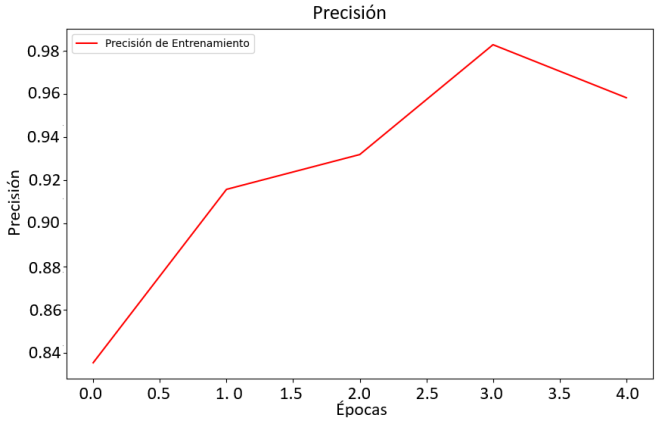

Figura 12. Grafo de precisión generado durante el entrenamiento de la segunda arquitectura propuesta.

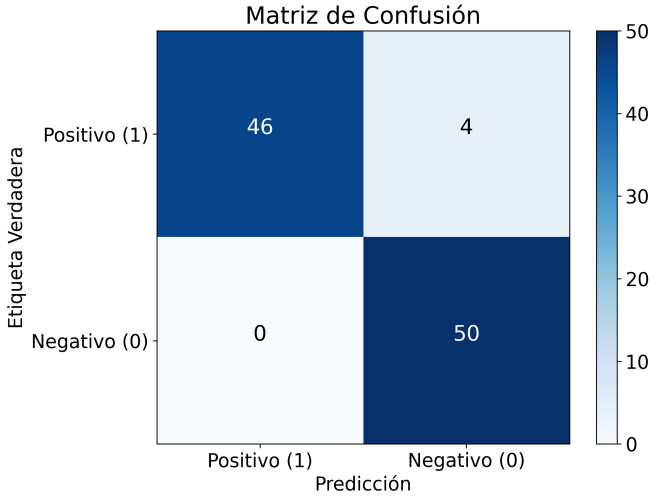

Figura 14. Matriz de Confusión generada tras probar la tercera arquitectura.

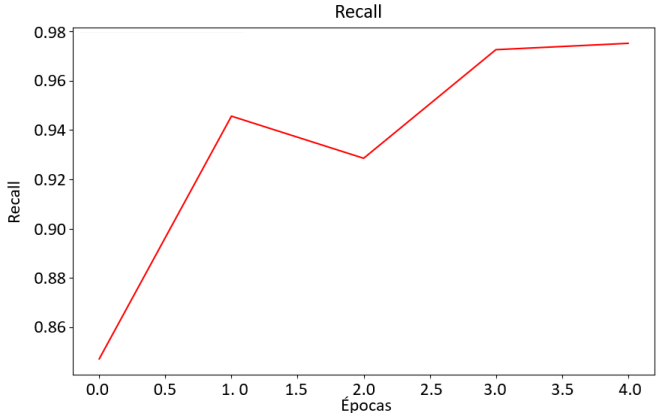

Figura 13. Grafo de recall generado durante el entrenamiento de la segunda arquitectura propuesta.

C. Arquitectura 3

Igual que los resultados anteriores, la matriz de confusión de la tercera arquitectura es casi perfecta, pero a diferencia de las anteriores, esta presenta 4 falsos negativos.

El entrenamiento con la arquitectura 3 dura 191 segundos, además para las pruebas dura un total de 21.7 segundos (ver cuadro I). Dado que el tiempo de entrenamiento es aproximadamente un cuarto del requerido por las otras implementaciones, se puede ajustar más parámetros e hiperparámetros sin consumir muchos recursos computacionales. En términos de complejidad temporal, esta implementación se presenta como la mejor opción propuesta.

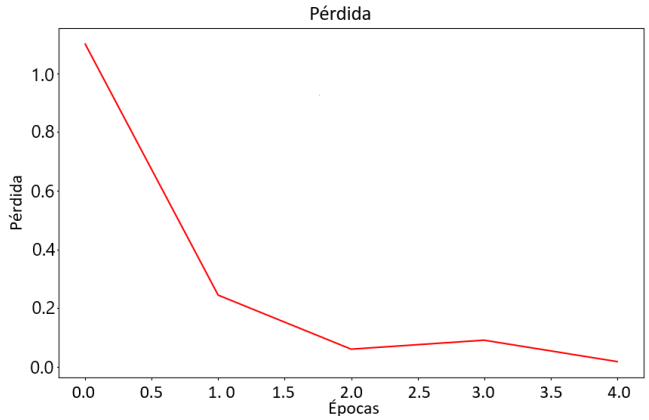

Figura 15. Grafo de pérdida generado durante el entrenamiento de la tercera arquitectura propuesta.

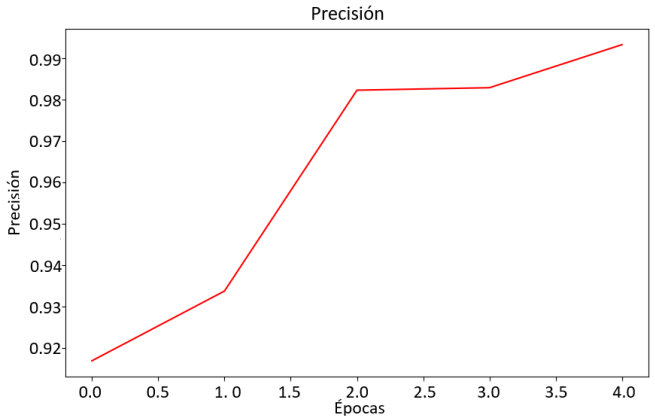

Figura 16. Grafo de precisión generado durante el entrenamiento de la tercera arquitectura propuesta.

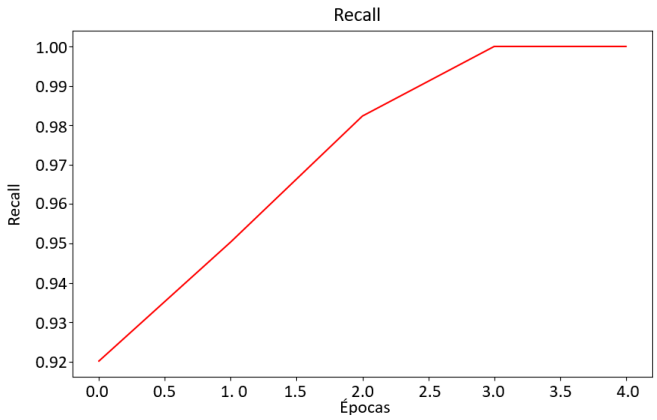

Figura 17. Grafo de recall generado durante el entrenamiento de la tercera arquitectura propuesta.

VIII. Conclusiones

A lo largo del desarrollo y tomando en cuenta los antecedentes, se ha demostrado que las CNN son altamente eficaces para identificar sonidos específicos de sirenas en entornos urbanos complejos. Esto no solo refuerza la precisión de estas redes en tareas de reconocimiento de audio, sino que también subraya su capacidad para ser adaptadas y optimizadas a contextos particulares, como el tráfico en San José, Costa Rica. Especialmente se denota la efectividad de la tercera arquitectura propuesta. Esta cuenta con una capa de MaxPooling, una capa de entrada, dos convolucionales con función de activación ReLu, seguido de una capa de aplanamiento.

Desde el marco teórico, el uso de redes neuronales convolucionales para el procesamiento de audio se ha consolidado como una técnica poderosa y versátil. La conversión de audios a espectrogramas ha permitido que las CNN identifiquen patrones acústicos complejos que son característicos de las sirenas de emergencia.

Los resultados obtenidos del proyecto responden directamente a los objetivos propuestos. En específico, se logró crear un modelo de detección de sirenas, por medio de las tres arquitecturas propuestas. Se evaluó el desempeño de estas y se definió la tercera como la preferida. Además de que se pudo crear un conjunto de datos a partir de recolecciones de campo y videos de internet para satisfacer las necesidades del modelo y el proyecto. Por lo que se logró la creación de un conjunto de datos específicos para el contexto costarricense y el desarrollo de arquitecturas de CNN optimizadas que permitieron alcanzar altos niveles de precisión en la detección de sirenas.

En conclusión, el proyecto establece una base para el uso de inteligencia artificial en la creación de sistemas de alerta para personas con discapacidad auditiva en Costa Rica. Por lo que las arquitecturas propuestas y la base de datos creada ofrecen resultados prometedores para usarlos y mejorarlos para futuras aplicaciones de la vida real. Para esto se recomienda, como mejoras del proyecto, hacer una recolección de datos más orientada al uso que se busca del sistema. Además de hacer una mayor investigación hacia las necesidades de las personas con discapacidades auditivas, serían los usuarios principales de este sistema.

Referencias

[1] R. A. Dobre, V. A. Niţă, A. Ciobanu, C. Negrescu, and D. Stanomir, "Low computational method for siren detection," in *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*. Brasov, Romania: IEEE, 2015, pp. 291–295.

[2] D. Pramanick, H. Ansar, H. Kumar, S. Pranav, R. Tengshe, and B. Fatimah, "Deep learning based urban sound classification and ambulance siren detector using spectrogram," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1–6.

[3] B. Fatimah, A. Preethi, V. Hrushikesh, A. S. B., and H. R. Kotion, "An automatic siren detection algorithm using fourier decomposition method and mfcc," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Kharagpur, India: IEEE, 2020, pp. 1–6.

[4] N. Ketkar and J. Moolayil, *Convolutional Neural Networks*. Berkeley, CA: Apress, 2021, pp. 197–242. [Online]. Available: https://doi.org/10.1007/978-1-4842-5364-9_6

[5] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020. [Online]. Available: https://doi.org/10.3102/1076998619872761

[6] M. E. Paz, G. R. Friedrich, and C. L. Galasso, "Procesamiento de señal visualizado sobre un espectrograma," *Elektron: ciencia y tecnología en la electrónica de hoy*, vol. 4, no. 1, pp. 35–39, 2020.

[7] Likebupt, "Smote - azure machine learning," Microsoft Learn, n.d., https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/smote?view=azureml-api-2.