CLASIFICACIÓN DE ARRITMIAS CARDIACAS A PARTIR DE SEÑALES FISIOLÓGICAS USANDO TÉCNICAS DE APRENDIZAJE PROFUNDO

Jesús David Artunduaga Vaquiro
Código 8974557
Owen Isaías Rocha Navarro
código 8980085

Proyecto Aplicado para optar al título de
Magister en Ciencia de Datos

Director
Julián Gil González

FACULTAD DE INGENIERÍA Y CIENCIAS
MAESTRÍA EN CIENCIA DE DATOS
SANTIAGO DE CALI, JUNIO 28 DE 2024

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

INTRODUCCIÓN

En este proyecto se propuso el desarrollo de un sistema de clasificación de señales de electrocardiogramas (ECG) utilizando técnicas de aprendizaje profundo. El objetivo general de este estudio es desarrollar un sistema de clasificación que utilice modelos de aprendizaje automático para la detección de arritmias cardiacas a partir de señales fisiológicas. Para lograrlo, se plantean los siguientes objetivos específicos: preparar un conjunto de datos de señales de electrocardiogramas (ECG) con arritmias cardiacas para el entrenamiento de modelos de aprendizaje profundo, implementar dos modelos de aprendizaje profundo para la detección de arritmias cardiacas, evaluar los modelos para medir su rendimiento y elaborar una interfaz gráfica de usuario que permita interactuar con el modelo de mejor rendimiento.

La metodología propuesta se basó en un enfoque sistemático y estructurado. Se inició con la preparación de un conjunto de datos relevantes de señales de electrocardiogramas con arritmias cardiacas, que incluyó la recopilación y limpieza de los datos. A continuación, se implementaron dos modelos de aprendizaje profundo, los cuales fueron ajustados y evaluados en cuanto a su rendimiento. Finalmente, se desarrolló una interfaz web para el modelo de clasificación de arritmias con el mejor rendimiento.

Como resultados esperados, se obtuvieron dos modelos de aprendizaje profundo entrenados y listos para su evaluación, así como una interfaz web que mejoró la experiencia del usuario final con el modelo que presentó mejor rendimiento. Además, se generó un informe final que documentó todo el proceso de investigación, incluyendo los resultados obtenidos, conclusiones y recomendaciones para futuras investigaciones.

Con este proyecto, se contribuye al campo de la detección de arritmias cardiacas mediante el uso de técnicas de aprendizaje profundo, brindando una herramienta que apoye el trabajo médico en el diagnóstico temprano de enfermedades cardíacas.

1. DEFINICIÓN DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

Según la Organización Mundial de la Salud (OMS), las enfermedades cardiovasculares (ECV) son la principal causa de muerte a nivel mundial [1] [2], cobrando aproximadamente 17,9 millones de vidas cada año [3]. En Colombia, en el año 2019, se registraron 128 muertes por cada 100.000 habitantes debido a estas enfermedades, según datos de la Organización Panamericana de la Salud (OPS) [4].

Dentro de las enfermedades cardiovasculares más comunes se encuentran las arritmias cardíacas, caracterizadas por alteraciones en el ritmo regular del corazón debido a un mal funcionamiento de los impulsos eléctricos que coordinan los latidos. Estas anomalías pueden resultar en una frecuencia cardíaca demasiado rápida (taquicardia), demasiado lenta (bradicardia) o irregular [5]. El diagnóstico de las arritmias se realiza a través de diversos métodos, siendo el electrocardiograma (ECG) uno de los más utilizados. El ECG registra gráficamente la actividad eléctrica del corazón y permite identificar diferentes ondas y segmentos [6].

El diagnóstico temprano de arritmias cardíacas es crucial en diferentes escenarios, como unidades de cuidados intensivos, salas de urgencias, períodos de recuperación hospitalaria e incluso en la vida diaria de las personas [7]. Actualmente, los cardiólogos o médicos generales revisan manualmente los datos de las señales fisiológicas, realizan el diagnóstico adecuado y comienzan a implementar planes de tratamiento [8]. Sin embargo, en entornos con alta carga de trabajo, donde se deben analizar múltiples señales fisiológicas, este proceso puede requerir el uso de mucho tiempo del personal capacitado, el cual podría emplearse en otras actividades médicas [9]. Además, en áreas con escasez de especialistas, los análisis pueden retrasarse hasta que lleguen profesionales calificados, lo que puede resultar en demoras en el diagnóstico y el inicio de los tratamientos [10]. Por lo tanto, es fundamental contar con una herramienta que realice el análisis automático de las señales fisiológicas a través de aprendizaje profundo, con el objetivo de asistir en el proceso de diagnóstico y aprovechar eficientemente los recursos médicos disponibles.

1.2. FORMULACIÓN DEL PROBLEMA

Se plantea la siguiente pregunta de investigación: ¿Cómo clasificar arritmias cardiacas empleando técnicas de aprendizaje profundo en el análisis de señales fisiológicas, específicamente el electrocardiograma (ECG)?

Para abordar este desafío, es necesario dar respuesta a una serie de preguntas claves que nos permitirán avanzar en la investigación. Estas preguntas incluyen:

- ¿Cuáles son los conjuntos de datos disponibles que contienen registros de electrocardiogramas (ECG) con arritmias cardiacas, y cómo podemos prepararlos para su uso en un modelo de aprendizaje profundo?
- ¿Qué tipo de arquitecturas de aprendizaje profundo son más apropiadas para desarrollar modelos capaces de clasificar arritmias cardiacas a partir de electrocardiogramas (ECG)?
- ¿Cómo podemos evaluar el rendimiento del modelo de aprendizaje profundo en la clasificación de arritmias cardiacas?
- ¿Cómo podemos diseñar una interfaz gráfica de usuario que optimice la interacción con el modelo de aprendizaje profundo que ha demostrado el mejor rendimiento?

2. OBJETIVOS DEL PROYECTO

2.1. OBJETIVO GENERAL

Clasificar arritmias cardiacas en señales de electrocardiograma (ECG), mediante el uso de técnicas de aprendizaje profundo.

2.2. OBJETIVOS ESPECÍFICOS

- Preparar un conjunto de datos de señales de electrocardiogramas (ECG) con arritmias cardiacas, para su uso en el entrenamiento de modelos de aprendizaje profundo.
- Implementar dos modelos de aprendizaje profundo para la detección de arritmias cardiacas a partir de electrocardiogramas (ECG).
- Evaluar los modelos para medir su rendimiento.
- Elaborar una interfaz gráfica de usuario para el modelo con el mejor rendimiento.

3. MARCO TEÓRICO Y ANTECEDENTES

3.1. MARCO TEÓRICO

3.1.1 Corazón

Desempeña un papel fundamental en el sistema circulatorio, ya que su principal función es bombear la sangre al resto del organismo. Se encuentra ubicado en el pecho, entre los pulmones, por detrás del esternón y por encima del diafragma. El corazón está compuesto por dos bombas separadas: el corazón derecho y el corazón izquierdo. Cada uno de ellos tiene dos cavidades, una aurícula y un ventrículo. Las aurículas, que tienen paredes delgadas, se encargan de llenar los ventrículos. Por su parte, los ventrículos, que poseen paredes más gruesas, contraen con mayor fuerza para impulsar la sangre hacia los pulmones (corazón derecho) o hacia los órganos periféricos (corazón izquierdo). Para prevenir el reflujo de la sangre desde los ventrículos hacia las aurículas, se encuentran presentes unas compuertas llamadas válvulas que separan estas cavidades. Asimismo, también existen válvulas que separan los ventrículos de las arterias principales. Estas válvulas son vitales para garantizar un flujo sanguíneo unidireccional y evitar el retroceso de la sangre [11].

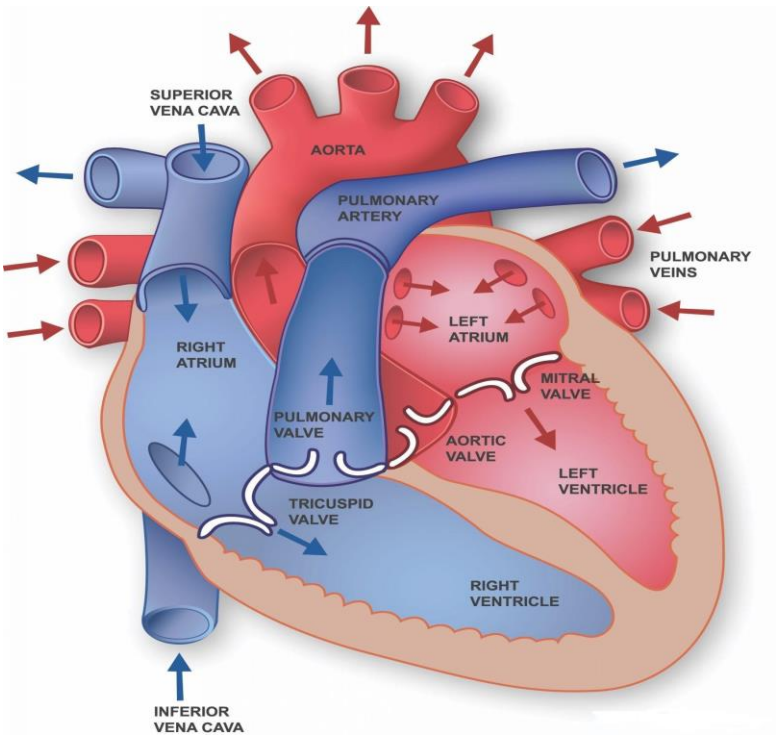

Fig. 1. Anatomía básica del corazón *[12]*

3.1.2 Arritmia Cardiaca

Se refiere a una condición en la que el ritmo normal del corazón se ve alterado. Se caracteriza por la presencia de latidos cardíacos irregulares, ya sea demasiado rápidos o demasiado lentos, que difieren del patrón regular de contracciones del corazón [5]. Estas alteraciones pueden afectar la frecuencia cardíaca, la regularidad de los latidos o la conducción eléctrica del corazón. Las arritmias cardíacas pueden surgir en distintas partes del corazón, como el nodo sinusal, el nodo atrioventricular, las vías de conducción o las fibras musculares del corazón. Existen diversos factores que pueden desencadenar estas arritmias, como enfermedades cardíacas, anomalías estructurales, desequilibrios en los niveles de electrolitos, consumo de ciertos medicamentos o sustancias, trastornos del sistema nervioso o influencias genéticas [13].

3.1.3 Tipos de Arritmias

Algunas arritmias pueden llegar a ser potencialmente mortales si no se tratan adecuadamente o si no se toman las medidas necesarias para restablecer un ritmo cardíaco normal [14]. A continuación, se describen algunas de ellas [5]:

- **Asistolia:** También conocida como paro cardíaco, es la falta total de actividad eléctrica en el corazón. En esta condición, el corazón deja de latir y no hay contracciones. Requiere atención médica inmediata y maniobras de reanimación cardiopulmonar (RCP).
- **Bradicardia extrema:** Se refiere a una frecuencia cardíaca anormalmente lenta, generalmente por debajo de 60 latidos por minuto. Si la bradicardia es extrema y causa una disminución significativa del flujo sanguíneo, puede provocar síntomas graves, como desmayos, mareos y dificultad para respirar.
- **Taquicardia extrema:** Se caracteriza por una frecuencia cardíaca anormalmente rápida, generalmente por encima de 100 latidos por minuto. Si la taquicardia es extrema y persiste durante períodos prolongados, puede aumentar el riesgo de complicaciones graves, como insuficiencia cardíaca o accidente cerebrovascular.
- **Taquicardia ventricular:** Es una arritmia en la cual los ventrículos del corazón late rápidamente y de manera descoordinada. Puede ser sostenida (ritmo cardíaco rápido y constante) o no sostenida (episodios breves de ritmo cardíaco rápido).

- **Aleteo/Fibrilación ventricular:** Son arritmias graves en las que los ventrículos del corazón presentan contracciones rápidas e incoordinadas, lo que impide el bombeo efectivo de la sangre.

3.1.4 Señales Fisiológicas

Las señales fisiológicas son datos biológicos medibles que reflejan el estado y funcionamiento de diversos sistemas del cuerpo humano. Estas señales son cruciales en el ámbito médico y de la investigación para monitorizar y diagnosticar diferentes condiciones de salud. Las señales fisiológicas pueden obtenerse de diversas fuentes, incluyendo la actividad eléctrica del corazón, la presión arterial, la respiración y otros parámetros vitales [15].

3.1.4.1 Electrocardiograma (ECG)

El electrocardiograma es una prueba médica no invasiva que registra la actividad eléctrica del corazón. Se utiliza para evaluar el ritmo y la función cardíaca, y es una herramienta diagnóstica ampliamente utilizada en cardiología. Durante un electrocardiograma, se colocan electrodos en la piel del pecho, los brazos y las piernas del paciente. Estos electrodos detectan las variaciones en el flujo de corriente eléctrica generadas por el corazón en cada ciclo cardíaco. La actividad eléctrica del corazón se registra en forma de ondas en un papel o una pantalla del monitor [16].

Las principales ondas que se visualizan en un electrocardiograma son [16]:

- **Onda P:** Representa la despolarización de las aurículas y se visualiza como una pequeña onda redonda antes del complejo QRS.
- **Complejo QRS:** Consta de una onda Q seguida de una onda alta y puntiaguda llamada onda R, y finaliza con una onda decreciente llamada onda S. Estas ondas indican la despolarización y la contracción de los ventrículos.
- **Segmento ST:** Es una sección plana y horizontal después del complejo QRS. Representa el período en el que los ventrículos están completamente despolarizados y listos para la siguiente fase del ciclo cardíaco.
- **Onda T:** Es una onda redondeada después del segmento ST y representa la repolarización de los ventrículos.

El electrocardiograma puede mostrar otros elementos y mediciones, como el intervalo PR, que indica la conducción eléctrica desde las aurículas hasta los ventrículos, y el intervalo QT, que mide la duración total de la actividad eléctrica ventricular [16] [17].

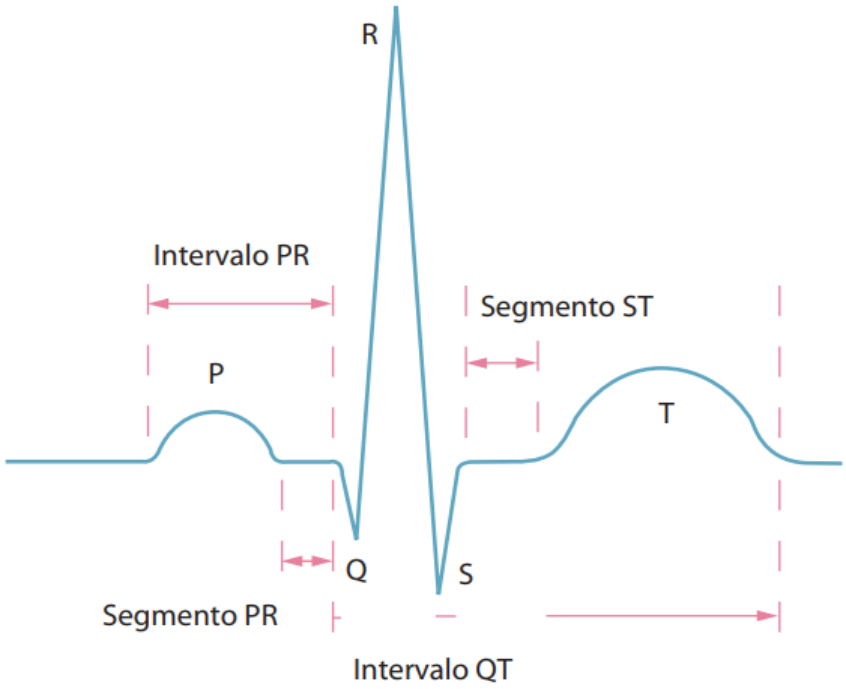

Fig. 2. Diferentes ondas del electrocardiograma *[17]*

Las derivaciones II y V del ECG son dos de las múltiples derivaciones utilizadas para obtener una visión más completa de la actividad eléctrica del corazón. La derivación II es comúnmente utilizada debido a su alta calidad en la representación de la onda P y el complejo QRS, lo cual es crucial para la detección de arritmias. La derivación V proporciona una perspectiva adicional que puede ayudar a identificar problemas específicos en diferentes partes del corazón [18].

3.1.4.2 Presión Arterial (ABP)

La presión arterial es la fuerza que ejerce la sangre contra las paredes de las arterias mientras el corazón la bombea. Se mide comúnmente en milímetros de mercurio (mmHg) y se registra como dos valores: la presión sistólica (máxima) y la presión diastólica (mínima). El monitoreo de la presión arterial es esencial para la evaluación de enfermedades cardiovasculares y la gestión de condiciones como la hipertensión [19].

3.1.4.3 Respiración (RESP)

La señal de respiración mide la frecuencia y el patrón respiratorio del paciente. Se obtiene mediante sensores que detectan el movimiento del pecho y el flujo de aire durante la inhalación y exhalación. Esta señal es importante para detectar problemas respiratorios como apnea del sueño, enfermedades pulmonares y para monitorizar el estado respiratorio en pacientes críticos [20].

3.1.4.4 Plethysmography (PLETH)

La pletismografía es una técnica utilizada para medir cambios en el volumen de sangre dentro de un órgano o una extremidad. La señal de pletismografía, comúnmente obtenida mediante un oxímetro de pulso, mide la saturación de oxígeno en la sangre y puede proporcionar información sobre la circulación periférica y la función respiratoria [21].

3.1.5 Técnicas de Filtrado

Para obtener una señal de ECG clara y precisa, es necesario aplicar técnicas de filtrado que eliminen el ruido y las interferencias. Dos filtros comunes utilizados en el procesamiento de señales de ECG son el Filtro de Kalman y el Filtro Butterworth [22].

- **Filtro de Kalman:** Es un algoritmo de filtrado recursivo que estima el estado de un sistema dinámico a partir de una serie de mediciones ruidosas. Es ampliamente utilizado en el procesamiento de señales de ECG para suavizar la señal y reducir el ruido aleatorio. Su capacidad para proporcionar estimaciones óptimas en tiempo real lo hace ideal para aplicaciones médicas donde se requiere precisión y rapidez [23].
- **Filtro Butterworth:** Es un tipo de filtro de paso bajo que tiene una respuesta en frecuencia lo más plana posible en la banda pasante. Es utilizado en el procesamiento de señales de ECG para eliminar el ruido de alta frecuencia y mejorar la visibilidad de las ondas importantes, como las ondas P, QRS y T. El Filtro Butterworth es preferido por su simplicidad y eficacia en la reducción de interferencias sin distorsionar la forma de la señal [24].

3.1.6 Detección de Picos R

La detección de los picos R es crucial para el análisis de la señal de ECG, ya que estos representan la despolarización de los ventrículos y son fundamentales para el cálculo de la

frecuencia cardíaca y la identificación de arritmias [25]. Uno de los algoritmos más conocidos y utilizados para la detección de picos R es el Algoritmo Pan y Tompkins, la cual es una técnica robusta para la detección de picos R en señales de ECG. Fue desarrollado en 1985 y se ha convertido en un estándar en el procesamiento de ECG debido a su alta precisión y eficiencia. El algoritmo consta de varias etapas, incluyendo filtrado, diferenciación, cuadratura e integración con una ventana deslizante, para resaltar los picos R y suprimir el ruido y las interferencias [22].

3.1.7 Aprendizaje Automático

El aprendizaje automático es un campo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender a partir de datos sin ser programadas explícitamente. Estos algoritmos utilizan métodos estadísticos para identificar patrones y hacer predicciones sobre nuevos datos [26].

Entre los métodos clásicos de aprendizaje automático se tienen [27]:

- **Regresión Lineal:** Es un método para modelar la relación entre una variable dependiente y una o más variables independientes. Se utiliza para predecir el valor de una variable continua.
- **Regresión Logística:** Es un modelo de regresión utilizada para problemas de clasificación binaria. Estima la probabilidad de que un evento ocurra dadas ciertas variables de entrada.
- **Máquinas de Vectores de Soporte (SVM):** Son un conjunto de algoritmos de aprendizaje supervisado utilizados para problemas de clasificación y regresión. Se basan en la idea de encontrar el hiperplano óptimo que mejor separa las clases en el espacio de características.
- **Árboles de Decisión:** Son modelos de aprendizaje automático que utilizan una estructura de árbol para representar y clasificar datos. Son especialmente útiles para problemas de clasificación y regresión.
- **K-means Clustering:** El algoritmo K-means es un método de clustering utilizado para agrupar datos en clusters o grupos basados en similitudes en las características. Es ampliamente utilizado en análisis de datos exploratorios y segmentación de clientes.

3.1.8 Aprendizaje Profundo

Es una rama del aprendizaje automático (machine learning) que se basa en algoritmos y modelos computacionales inspirados en la estructura y función del cerebro humano. Las redes neuronales artificiales profundas son una de las técnicas mas robustas para aprender y extraer características complejas de conjuntos de datos, permitiendo así el procesamiento y análisis de información de manera más avanzada [28]. Algunos tipos de redes neuronales comunes son la Red Neuronal Convolucional (CNN) y la Red Neuronal Recurrentes (RNN).

3.1.8.1 Red Neuronal Convolucional (CNN)

Son altamente eficaces en el procesamiento de datos con estructura de cuadrícula, como imágenes. Su arquitectura se compone de capas convolucionales que aplican filtros para extraer características locales de la imagen. Estas capas son seguidas por capas de agrupación (pooling) que reducen la dimensionalidad de la representación, contribuyendo así a una mejora en la eficiencia computacional y evitando el sobreajuste [28].

El modelo típico de una CNN consiste en varias capas convolucionales seguidas de capas de agrupación y finalmente capas completamente conectadas para la clasificación. Cada capa convolucional aplica un conjunto de filtros convolucionales a la entrada para extraer características relevantes. Las capas de agrupación reducen la dimensionalidad de las características para mejorar la eficiencia computacional y evitar el sobreajuste [28] .

Las aplicaciones de este tipo de redes han revolucionado el campo de la visión por computadora y se extienden a diversos ámbitos, como el reconocimiento de objetos en imágenes médicas, la detección de fraudes en transacciones financieras y el reconocimiento facial en sistemas de seguridad.

3.1.8.2 Red Neuronal Recurrente (RNN)

Son ideales para modelar datos secuenciales, como texto o series temporales, debido a su capacidad para mantener y utilizar información a lo largo del tiempo. A diferencia de las CNN, las RNN tienen conexiones retroalimentadas que les permiten capturar dependencias temporales en los datos. Este tipo de arquitectura ha encontrado aplicaciones en una variedad de tareas,

incluyendo el procesamiento del lenguaje natural, la generación de texto y la traducción automática [29].

El modelo típico de una RNN consta de una estructura recurrente que permite que la información se propague hacia adelante en el tiempo. Sin embargo, las RNN tradicionales tienen dificultades para capturar dependencias a largo plazo debido al problema de desvanecimiento / explosión del gradiente. El modelo de LSTM (Long Short-Term Memory) aborda este problema al introducir unidades de memoria que pueden recordar información relevante durante períodos prolongados [29].

3.1.8.3 Tipos de capas en redes neuronales

- **Capa Convolucional:** Son fundamentales en las redes neuronales convolucionales (CNN). Utilizan filtros (o kernels) para realizar convoluciones sobre la entrada, permitiendo la detección de características locales en los datos, como bordes en imágenes o patrones en señales. Cada filtro se desliza sobre la entrada y calcula productos punto entre el filtro y secciones de la entrada. Las características aprendidas se combinan a lo largo de múltiples filtros para capturar representaciones complejas de los datos. La función de activación ReLU (Rectified Linear Unit) se utiliza frecuentemente para introducir no linealidad al modelo, permitiendo que la red neuronal aprenda relaciones complejas [30].
- **Capa de Max-Pooling:** La capa de max-pooling reduce la dimensionalidad de las características extraídas por las capas convolucionales seleccionando el valor máximo en cada región de la entrada. Esto ayuda a disminuir la cantidad de parámetros y el costo computacional, además de proporcionar invariancia a pequeñas traducciones en la entrada. El max-pooling también ayuda a evitar el sobreajuste al reducir la complejidad del modelo. El tamaño de pool define la dimensión de la región sobre la cual se calcula el valor máximo, normalmente configurado en 2x2 [31].
- **Capa de Flatten:** La capa de flatten convierte las salidas multidimensionales de las capas convolucionales en un vector unidimensional. Esta transformación es necesaria para conectar las capas convolucionales con las capas densas, que esperan una entrada unidimensional. Al aplanar las características extraídas, esta capa prepara los datos para ser procesados por las capas densas posteriores, facilitando la integración de información

compleja en un formato que puede ser utilizado para la toma de decisiones [31].

- **Capa Densa:** Conocidas como capas completamente conectadas, son aquellas en las que cada neurona está conectada a todas las neuronas de la capa anterior. Estas capas son responsables de combinar las características extraídas y realizar la clasificación o regresión final. Las neuronas en una capa densa realizan una combinación lineal de las entradas seguida de una función de activación. La función de activación ReLU es comúnmente usada en capas ocultas, mientras que funciones como softmax se utilizan en la capa de salida para tareas de clasificación [32].
- **Capa de Dropout:** Es una técnica de regularización utilizada para prevenir el sobreajuste en redes neuronales. Durante el entrenamiento, apaga aleatoriamente una fracción de las neuronas en la capa a la que se aplica, lo que obliga a la red a no depender excesivamente de ciertas neuronas. La tasa de dropout define la proporción de neuronas que se desactivan durante el entrenamiento (por ejemplo, 50%). Esta capa mejora la capacidad de generalización de la red y reduce el riesgo de sobreajuste, asegurando que el modelo sea robusto y pueda manejar datos no vistos de manera efectiva [33].
- **Capa LSTM:** Las Long Short-Term Memory (LSTM) son un tipo especial de red neuronal recurrente (RNN) que puede aprender dependencias a largo plazo en datos secuenciales. Las LSTM están diseñadas para recordar información durante largos periodos y evitar el problema del desvanecimiento del gradiente que afecta a las RNN tradicionales. Cada unidad LSTM tiene una estructura compleja con puertas de entrada, olvido y salida que controlan el flujo de información. Esto permite que las LSTM sean extremadamente efectivas para tareas como el modelado de series temporales, traducción automática y procesamiento de lenguaje natural [33].
- **Capa GRU:** Las Gated Recurrent Units (GRU) son otra variante de las redes neuronales recurrentes diseñadas para resolver problemas similares a los de las LSTM. Las GRU simplifican la estructura interna de las LSTM combinando las puertas de entrada y olvido en una sola puerta de actualización y la puerta de salida en una puerta de reinicio. Esta simplificación hace que las GRU sean menos computacionalmente intensivas mientras siguen siendo muy efectivas para capturar dependencias a largo plazo en datos secuenciales. Las GRU son especialmente útiles en aplicaciones donde los recursos computacionales son

limitados y se necesita una implementación eficiente [33].

3.1.9 Ajuste de Hiperparámetros

El proceso de ajuste de hiperparámetros es esencial en el desarrollo de modelos de aprendizaje automático, ya que implica encontrar la configuración óptima del modelo para maximizar su rendimiento predictivo. Para lograr esto, se recurre a diversas técnicas de ajuste de hiperparámetros. Entre las más comunes destacan:

- **Búsqueda en Cuadrícula (Grid Search):** Este método implica definir un conjunto de valores para cada hiperparámetro y evaluar todas las combinaciones posibles utilizando validación cruzada. Es útil cuando el espacio de búsqueda de los hiperparámetros es manejable, pero puede volverse computacionalmente costoso en espacios grandes o altamente dimensionales.
- **Búsqueda Aleatoria (Random Search):** Esta alternativa selecciona aleatoriamente combinaciones de hiperparámetros para su evaluación. Aunque puede parecer contraintuitivo, a menudo supera a la búsqueda en cuadrícula en eficiencia y efectividad, ya que puede explorar de manera más eficiente el espacio de hiperparámetros.
- **Optimización Bayesiana:** Utiliza técnicas bayesianas para buscar de manera eficiente el espacio de hiperparámetros. Se basa en la construcción de un modelo probabilístico de la función objetivo y su optimización para encontrar la mejor configuración de hiperparámetros. Es especialmente útil en espacios continuos y cuando el número de evaluaciones de la función objetivo es limitado [34].
- **Entropía Cruzada:** La entropía cruzada es una medida utilizada en teoría de la información y el aprendizaje automático para cuantificar la diferencia entre dos distribuciones de probabilidad, generalmente aplicada como una función de pérdida en modelos de clasificación. Evalúa cómo la distribución de etiquetas verdaderas se compara con las predicciones del modelo, penalizando fuertemente las predicciones incorrectas. Matemáticamente, se define como la suma del producto de las probabilidades verdaderas y el logaritmo de las probabilidades predichas. Al minimizar la entropía cruzada durante el entrenamiento, los algoritmos de optimización ajustan los parámetros del modelo para mejorar su precisión y capacidad de generalización, haciendo que este concepto sea crucial para la efectividad de los modelos de deep learning y machine learning [35].

3.1.10 Métricas de evaluación

En la evaluación de modelos de aprendizaje automático, es fundamental utilizar métricas adecuadas que permitan medir su rendimiento y generalización. A continuación, se presentan las métricas de evaluación más comúnmente utilizadas en este contexto:

- **Accuracy (Precisión):** Mide la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones. Se calcula dividiendo el número de predicciones correctas (verdaderos positivos (TP) y verdaderos negativos (TN)) entre el total de predicciones. Los verdaderos positivos representan las muestras positivas que fueron correctamente clasificadas, los verdaderos negativos son las muestras negativas que fueron correctamente clasificadas, los falsos positivos (FP) son las muestras negativas que fueron incorrectamente clasificadas como positivas, y los falsos negativos (FN) son las muestras positivas que fueron incorrectamente clasificadas como negativas [36].

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

- **Matriz de Confusión:** Es una herramienta que describe el rendimiento de un modelo de clasificación en términos de las clases verdaderas y predichas. Proporciona una visión detallada de los aciertos y errores del modelo, permitiendo analizar su desempeño en diferentes aspectos [36]. La matriz de confusión se representa de la siguiente manera:

	Clase Positiva	*Clase Negativa*
Clase Positiva	*Verdaderos Positivos (TP)*	*Falsos Negativos (FN)*
Clase Negativa	*Falsos Positivos (FP)*	*Verdaderos Negativos (TN)*

- **Recall (Sensibilidad):** Es una métrica de evaluación utilizada en aprendizaje automático y minería de datos para medir la capacidad de un modelo de identificar correctamente las instancias positivas de una clase [37]. Específicamente, el recall se define como la proporción de verdaderos positivos sobre el total de verdaderos positivos y falsos negativos. Matemáticamente, el recall se expresa como:

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

El recall es especialmente importante en situaciones donde es crucial identificar la mayor

cantidad posible de casos positivos, como en diagnósticos médicos, detección de fraudes y sistemas de seguridad. Un alto valor de recall indica que el modelo tiene una alta capacidad para detectar instancias positivas, aunque esto puede venir a costa de un mayor número de falsos negativos. Por lo tanto, el recall se utiliza frecuentemente junto con la precisión (precisión) para proporcionar una visión más completa del rendimiento del modelo, a menudo combinándose en la métrica F1 para equilibrar ambos aspectos [38].

- **F1-Score:** Es una medida de precisión que combina tanto la precisión como el recall del modelo en un solo valor. Se calcula como la media armónica entre la precisión y el recall, lo que proporciona una métrica equilibrada entre ambas. El F1-Score alcanza su mejor valor en 1 (indicando una clasificación perfecta) y su peor valor en 0 (indicando un rendimiento deficiente del modelo) [36].

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

3.1.11 Herramientas y bibliotecas utilizadas en el análisis de datos y modelado

- **NumPy:** Es una biblioteca fundamental para la computación científica en Python. Fue desarrollada originalmente por Travis Oliphant en 2005 como una extensión del proyecto Numeric. NumPy proporciona soporte para arreglos y matrices multidimensionales junto con una colección de funciones matemáticas de alto nivel para operar con ellos [39]. Esta biblioteca es la base para muchas otras bibliotecas científicas debido a su eficiencia y flexibilidad en el manejo de datos numéricos. Se utiliza para realizar operaciones matemáticas y estadísticas sobre grandes conjuntos de datos, facilitando el trabajo con matrices y arreglos multidimensionales. La versatilidad y el rendimiento de NumPy han hecho que sea una herramienta imprescindible en la computación científica y en el desarrollo de modelos de aprendizaje automático [40].
- **Pandas:** Es una biblioteca de código abierto en Python, creada por Wes McKinney en 2008. Proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar. Pandas se construyó sobre NumPy y ofrece estructuras de datos como Series (unidimensional) y DataFrame (bidimensional), que permiten realizar operaciones

complejas de manipulación y análisis de datos de manera eficiente [41]. Es ampliamente utilizada en la ciencia de datos para tareas de limpieza, manipulación y análisis de datos. Su capacidad para manejar grandes conjuntos de datos y realizar operaciones avanzadas de manera eficiente ha transformado significativamente el análisis de datos en Python, convirtiéndose en una herramienta esencial para científicos de datos y analistas [42].

- **SciPy:** Es una biblioteca de código abierto que utiliza NumPy para proporcionar rutinas numéricas y algorítmicas adicionales. Fue desarrollada para ser utilizada en matemáticas, ciencia e ingeniería, ofreciendo funciones para la integración, optimización, álgebra lineal y procesamiento de señales, entre otras. SciPy extiende las capacidades de NumPy al proporcionar algoritmos y herramientas adicionales que son esenciales para tareas más avanzadas de computación científica y análisis de datos complejos [43].
- **TensorFlow:** Es una plataforma de código abierto para el aprendizaje automático desarrollada por Google Brain. Proporciona un amplio soporte para construir y entrenar modelos de machine learning y deep learning. TensorFlow permite la creación de redes neuronales avanzadas y es conocido por su eficiencia y flexibilidad en el manejo de grandes volúmenes de datos y tareas computacionales intensivas [44]. Desde su lanzamiento en 2015, ha sido ampliamente adoptado en la investigación y la industria debido a su capacidad para escalar desde el entrenamiento de modelos en dispositivos individuales hasta implementaciones distribuidas en grandes clústeres de servidores [45].
- **Keras:** Es una API de alto nivel para redes neuronales, diseñada para facilitar la construcción y el entrenamiento de modelos de deep learning. Proporciona una interfaz sencilla y modular que permite a los desarrolladores crear rápidamente prototipos de modelos complejos. Keras está integrada en TensorFlow, lo que le permite aprovechar las capacidades avanzadas de esta plataforma. Utilizada para construir y entrenar modelos de deep learning de manera eficiente y fácil de entender, Keras es ideal tanto para principiantes como para investigadores avanzados. Su diseño amigable y su integración con TensorFlow han facilitado la adopción del deep learning en una amplia variedad de aplicaciones [46].
- **Scikit-learn:** Es una biblioteca de aprendizaje automático en Python, construida sobre NumPy, SciPy y matplotlib. Proporciona herramientas simples y eficientes para el análisis de datos y machine learning, incluyendo clasificación, regresión, agrupamiento y reducción

de dimensionalidad. Es utilizada ampliamente para tareas de aprendizaje supervisado y no supervisado en ciencia de datos. La simplicidad y efectividad de Scikit-learn la han convertido en una herramienta estándar para la implementación de algoritmos de machine learning, facilitando el desarrollo rápido de modelos predictivos y análisis de datos [46].

- **Matplotlib:** Es una biblioteca de gráficos en Python que permite generar visualizaciones estáticas, animadas e interactivas. Fue desarrollada por John D. Hunter en 2003 y se ha convertido en una herramienta fundamental para la visualización de datos en la comunidad de ciencia de datos. Se utiliza para crear una amplia variedad de gráficos, como líneas, barras, histogramas y dispersión, facilitando la visualización de datos y resultados [47]. La capacidad de Matplotlib para generar visualizaciones de alta calidad ha hecho que sea una opción preferida para científicos y analistas que necesitan comunicar sus hallazgos de manera efectiva [48].
- **Keras Tuner:** Es una biblioteca de ajuste de hiperparámetros para modelos de machine learning. Facilita el proceso de búsqueda de la mejor configuración de hiperparámetros para optimizar el rendimiento del modelo. Utilizada para realizar optimización de hiperparámetros en modelos de deep learning, mejorando su rendimiento y precisión. Esta herramienta permite a los desarrolladores experimentar con diferentes configuraciones de modelos de manera eficiente, lo que es crucial para lograr modelos de alta calidad y rendimiento en aplicaciones de machine learning [46].
- **Streamlit:** Es una biblioteca de código abierto que permite crear aplicaciones web interactivas y personalizadas para machine learning y ciencia de datos en Python. Fue lanzada en 2019 y se ha vuelto popular por su simplicidad y rapidez para convertir scripts de Python en aplicaciones web. Utilizada para desarrollar dashboards y aplicaciones web interactivas que permiten a los usuarios interactuar con modelos y datos de manera intuitiva, Streamlit ha facilitado la creación de prototipos y la presentación de resultados de manera accesible y visualmente atractiva [49].
- **Pyngrok:** Es una biblioteca que proporciona una interfaz de Python para ngrok, una herramienta que crea túneles seguros desde una URL pública hacia una aplicación local. Es especialmente útil para exponer aplicaciones web locales a internet de manera segura. Utilizada para hacer accesibles las aplicaciones web locales a través de internet sin necesidad

de configuraciones complejas de red, Pyngrok ha simplificado significativamente el desarrollo y la prueba de aplicaciones web en entornos de desarrollo [50].

- **WFDB:** Es una biblioteca para el manejo y procesamiento de bases de datos de señales fisiológicas. Facilita la lectura, escritura y manipulación de estos datos, siendo particularmente útil en el ámbito de la investigación biomédica. Utilizada para el manejo y procesamiento de señales fisiológicas en formatos específicos utilizados en bases de datos médicas, WFDB proporciona herramientas esenciales para investigadores que trabajan con datos fisiológicos y necesitan realizar análisis complejos de manera eficiente [51].
- **H5py:** Es una interfaz de Python para el formato de datos HDF5 (Hierarchical Data Format), que permite almacenar grandes cantidades de datos de manera organizada y eficiente. Utilizada para el almacenamiento y manipulación de grandes volúmenes de datos, facilitando el acceso y procesamiento eficiente de estos datos en aplicaciones científicas y de ingeniería, H5py es fundamental en proyectos que requieren el manejo de datos masivos y la organización estructurada de la información [52].

3.2. ANTECEDENTES

A continuación, se presentarán algunas investigaciones que se han realizado en busca de métodos de detección automática de enfermedades cardiovasculares.

Villagrán Fuentes, C. A, "Clasificación automática de latidos de un electrocardiograma utilizando aprendizaje profundo". Tesis de maestría, Universidad de Concepción, Chile, 2017 [53]**.**

En la investigación "Clasificación automática de latidos de un electrocardiograma utilizando aprendizaje profundo", el autor aborda la problemática de las enfermedades cardiovasculares mediante la implementación de cuatro arquitecturas de redes neuronales convolucionales (CNN) para clasificar latidos cardíacos utilizando la base de datos de arritmias del MIT-BIH. Los datos fueron preprocesados y divididos en conjuntos de entrenamiento (80%) y prueba (20%) bajo un enfoque inter-paciente, empleando técnicas de balanceo de clases. Las CNN, desarrolladas en Python 2.7 con TensorFlow, incluyeron tres capas convolucionales, capas de agrupación y una capa softmax para la clasificación. Los modelos lograron precisiones de hasta 94.4%, destacando

la arquitectura de tres capas convolucionales como la más efectiva. Este estudio demuestra la eficacia de las CNN en la detección precisa de arritmias, sugiriendo su potencial para aplicaciones médicas y la importancia de mejorar el balanceo de clases y explorar el uso de ECG de dispositivos vestibles para futuras investigaciones.

F. A. Hernández Rueda, "Clasificación de registros ECG de corta duración usando redes neuronales recurrentes", Tesis de maestría, Universidad Pontificia Bolivariana, Bucaramanga, Colombia, 2020 [54].

En la investigación "Clasificación de registros ECG de corta duración usando redes neuronales recurrentes" se proponen tres modelos de clasificación para realizar la detección de los ritmos cardíacos. El primer modelo se basa en una red neuronal artificial (RNA) y utiliza características extraídas de la serie de intervalos del electrocardiograma (ECG). Los otros dos modelos se basan en redes neuronales recurrentes del tipo LSTM bidireccionales. Estos modelos se entrenaron utilizando diferentes series de tiempo. Los resultados preliminares muestran un desempeño del 64% en el primer modelo, 70.71% y 82.99% en los dos modelos basados en redes neuronales recurrentes. Estos resultados demuestran el potencial de las técnicas de aprendizaje automático, como las redes neuronales recurrentes, para la clasificación precisa de los ritmos cardíacos en registros ECG de corta duración.

Bergamini, M. L., & Liberczuk, S. J, "Detección de patologías en señales biomédicas mediante técnicas de machine learning. In XXII Workshop de Investigadores en Ciencias de la Computación, 2020 [55]**.**

Este estudio aborda la importancia del procesamiento de señales biomédicas, en particular el electrocardiograma (ECG), en el diagnóstico temprano y la prevención de enfermedades cardíacas. El proyecto se centra en el desarrollo y aplicación de algoritmos avanzados para el análisis, modelado, clasificación y segmentación de señales electrocardiográficas que pueden aplicarse en tiempo real para apoyar la detección temprana de eventos patológicos. Se proponen algoritmos de procesamiento de ECG con un enfoque Bayesiano para sintonizar los parámetros de un modelo dinámico, lo que permite la síntesis de señales de ECG durante procesos de isquemia e infarto. Además, se aplican técnicas de machine learning para configurar un sistema de asistencia al diagnóstico automático de patologías. La investigación se lleva a cabo en colaboración con el

Centro de Altos Estudios en Tecnología Informática (CAETI) de la Universidad Abierta Interamericana, el Instituto Argentino de Matemática Alberto P. Calderón (IAM) dependiente del CONICET, y el Instituto de Ingeniería Biomédica (IIBM) de la Universidad de Buenos Aires. Entre los logros obtenidos, se destaca un algoritmo eficiente para la estimación de parámetros del modelo de la actividad eléctrica del corazón utilizando optimización global heurística basada en simulación de Monte Carlo, la clasificación de latidos con support vector machine (SVM), y un algoritmo de filtrado con enfoque bayesiano que ha mostrado alta performance en diferentes niveles de SNR. Los resultados esperados incluyen la caracterización de señales ECG patológicas en el espacio de los parámetros del modelo, un nuevo sistema de representación de señales para detectar patrones.

J. Blanco Prieto, "Detección, clasificación e interpretación de patologías de la conducción cardíaca mediante el uso de técnicas de deep learning", Tesis de maestría, Universidad de Oviedo, 2021 [56]**.**

En la tesis de maestría "Detección, clasificación e interpretación de patologías de la conducción cardiaca mediante el uso de técnicas de deep learning" se aborda el desafío de mejorar la detección, clasificación e interpretabilidad de patologías cardiacas mediante el uso de técnicas de deep learning. Se propuso desarrollar un algoritmo inteligente basado en redes neuronales convolucionales de una dimensión (CNN1D) para realizar una clasificación automática de patologías cardiacas a partir de señales de ECG. El estudio se divide en tres análisis. En el primero, se utilizó un modelo CNN1D sin extracción previa de características, lo que resultó en una alta precisión en la clasificación de patologías, pero presentó dificultades para ser interpretado por los especialistas. En el segundo análisis, se aplicaron técnicas de extracción de características previas, pero se observó una disminución en la precisión y problemas en la detección de ciertas patologías. En el tercer análisis, al excluir las patologías problemáticas, se obtuvieron resultados satisfactorios con una precisión del 83% en entrenamiento y 82% en prueba. Se utilizaron técnicas de procesamiento de señales para identificar características distintivas. Además, se aplicaron técnicas de interpretación, como CAM y SHAP, que demostraron ser efectivas para aumentar la confianza y la aplicabilidad de los modelos de deep learning en entornos médicos.

C. F. Patiño Zambrano, "Dispositivo vestible inteligente para la generación de alertas tempranas de eventos cardiovasculares de riesgo", Tesis de maestría, Universidad EIA, Envigado, 2022 [57]**.**

En el proyecto "Dispositivo Vestible Inteligente para la Generación de Alertas Tempranas de Eventos Cardiovasculares de Riesgo" se desarrolló un dispositivo que emplea la variabilidad de la frecuencia cardiaca (HRV) como predictor de riesgos cardiovasculares. Se implementaron modelos de machine learning supervisado, como árboles de decisión, clasificador Naïve Bayes, regresión logística, KNN y Random Forest, para la detección de eventos anómalos en las variables fisiológicas. La implementación del algoritmo se realizó en un dispositivo portátil de bajo consumo, utilizando técnicas de Tiny Machine Learning. Los resultados mostraron una alta precisión y efectividad en la detección de eventos cardiovasculares de riesgo.

De los trabajos citados se puede concluir que existen diversos enfoques para la clasificación de arritmias cardiacas mediante técnicas de aprendizaje profundo. Los estudios revisados destacan el uso de arquitecturas de redes neuronales, como las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN), junto con técnicas de procesamiento de señales y balanceo de clases para mejorar la precisión y la robustez de los modelos.

Lo que distingue y hace innovador al proyecto propuesto es la implementación y comparación de dos arquitecturas de redes neuronales diferentes: CNN y RNN, ajustadas mediante técnicas de búsqueda aleatoria de hiperparámetros, para detectar y clasificar arritmias en señales de ECG. Además, se utilizó un conjunto de datos preprocesado y etiquetado específicamente para abordar tanto la veracidad de la alarma como el tipo de arritmia, lo cual mejora significativamente el enfoque general del sistema.

Estas características no solo potencian la precisión y efectividad del sistema propuesto, sino que también facilitan su aplicabilidad mediante una interfaz gráfica de usuario, ofreciendo un recurso de apoyo en la detección y clasificación de arritmias cardiacas.

4. DESARROLLO DEL PROYECTO

4.1 METODOLOGÍA

En esta sección se describe la metodología aplicada en cuatro etapas para alcanzar los objetivos establecidos en este proyecto aplicado.

- **Preparación de los datos:** Se obtuvieron las señales fisiológicas del electrocardiograma (ECG) de la competencia *Reducing False Arrhythmia Alarms in the ICU: The PhysioNet/Computing in Cardiology Challenge 2015* [14], accesibles a través del portal web PhysioNet. Estas señales se almacenaron en archivos con extensión .mat y .hea, formatos binarios utilizados por MATLAB. Para su procesamiento en Python, se utilizaron funciones de la biblioteca Scipy para la importación de datos. Posteriormente, se llevó a cabo un preprocesamiento de las señales ECG, que incluyó la eliminación de ruido, la normalización, la implementación de un detector de picos R y la segmentación de las señales para su posterior análisis.
- **Implementación de modelos:** Se seleccionaron arquitecturas de red neuronal convolucional (CNN) y red neuronal recurrente (RNN) para la implementación de los modelos. Se entrenaron ambas arquitecturas utilizando los datos preprocesados y se ajustaron los hiperparámetros correspondientes buscando optimizar su rendimiento. El ajuste de hiperparámetros se realizó mediante la técnica de búsqueda aleatoria utilizando Keras Tuner, optimizando el número de filtros en las capas convolucionales, el número de unidades en las capas LSTM, el número de unidades en las capas densas y la tasa de dropout.
- **Evaluación:** Los modelos entrenados se evaluaron utilizando un conjunto de métricas que incluyen precisión, sensibilidad, recall y F1-Score. Estas métricas permiten una evaluación completa del rendimiento de los modelos en la clasificación de alarmas y arritmias.
- **Interfaz web:** Se diseñó una interfaz web para el modelo con el mejor rendimiento, con el objetivo de facilitar su uso y comprensión. La interfaz de usuario fue desarrollada utilizando la biblioteca Streamlit, permitiendo una interacción intuitiva y accesible para usuarios sin conocimientos técnicos profundos. La interfaz permite cargar archivos de señales de ECG, procesarlos y visualizar los resultados de las predicciones, indicando si la alarma es verdadera o falsa y el tipo de arritmia detectada.

4.2 PREPARACIÓN DE LOS DATOS

4.2.1 Entendimiento de los datos

Los datos utilizados en este estudio provienen de la competencia *Reducing False Arrhythmia Alarms in the ICU: The PhysioNet/Computing in Cardiology Challenge 2.015* [14] sobre la reducción de falsas alarmas de arritmia en la Unidad de Cuidados Intensivos (UCI). Esta competencia proporcionó un conjunto de 1.250 segmentos de datos fisiológicos de la UCI asociados con alarmas críticas de arritmia.

Estos datos fueron recopilados de unidades de cuidados intensivos en cuatro hospitales en los Estados Unidos y Europa, representando a tres importantes fabricantes de equipos de monitoreo de UCI [14]. Sin embargo, debido a restricciones en la disponibilidad de datos, solo es posible acceder y descargar un total de 750 señales de electrocardiogramas (ECG) para el desarrollo de este estudio.

La etiqueta de cada alarma en los datos se estableció mediante un proceso de revisión por expertos, donde se determinó si la alarma era falsa o verdadera, así como el tipo específico de arritmia presente en la señal de electrocardiograma [14]. Para construir el conjunto de datos de alarmas verdaderas y falsas, un equipo de expertos inspeccionó visualmente el registro de formas de onda en el momento de cada alarma. Cada anotador trabajó de forma independiente y etiquetó la alarma como verdadera o falsa después de revisar 15 segundos de formas de onda antes de la alarma y 5 segundos después de ella. Además, identificaron el tipo de arritmia presente en cada señal, incluyendo asistolia, bradicardia extrema, taquicardia extrema, taquicardia ventricular, y aleteo/fibrilación ventricular. Para que una alarma fuera incluida en el conjunto de datos, al menos dos anotadores tenían que estar de acuerdo en que la alarma era verdadera o falsa.

TABLA I
FRECUENCIAS CARDÍACAS POR TIPO DE ARRITMIA Y TIPO DE ALARMA

Tipo de Arritmia	Tipo de alarma	
	Falsa	Verdadera
Asistolia	100	22
Bradicardia	43	46
Taquicardia	9	131
Aleteo/fibrilación ventricular	52	6
Taquicardia Ventricular	252	89
Total	456	294
	750	

Nota: La tabla muestra las frecuencias cardíacas por tipo de arritmia y tipo de alarma.

Cada segmento de datos tiene una duración de 5 minutos para análisis en tiempo real, finalizando en el momento de la alarma. Adicionalmente, al 50% de los registros se les proporcionaron 30 segundos adicionales de datos después de que se activara la alarma para análisis retrospectivo.

TABLA II
DURACIÓN Y CANTIDAD DE SEGMENTOS DE DATOS

Duración (segundos)	Cantidad
330	375
300	375
Total	750

Nota: Duración y cantidad de segmentos de datos registrados.

El siguiente gráfico presenta la frecuencia de duración de registros por tipo de arritmia y tipo de alarma, proporcionando una visualización de cómo se distribuyen los segmentos de datos en función de su duración en segundos para cada tipo de arritmia y alarma.

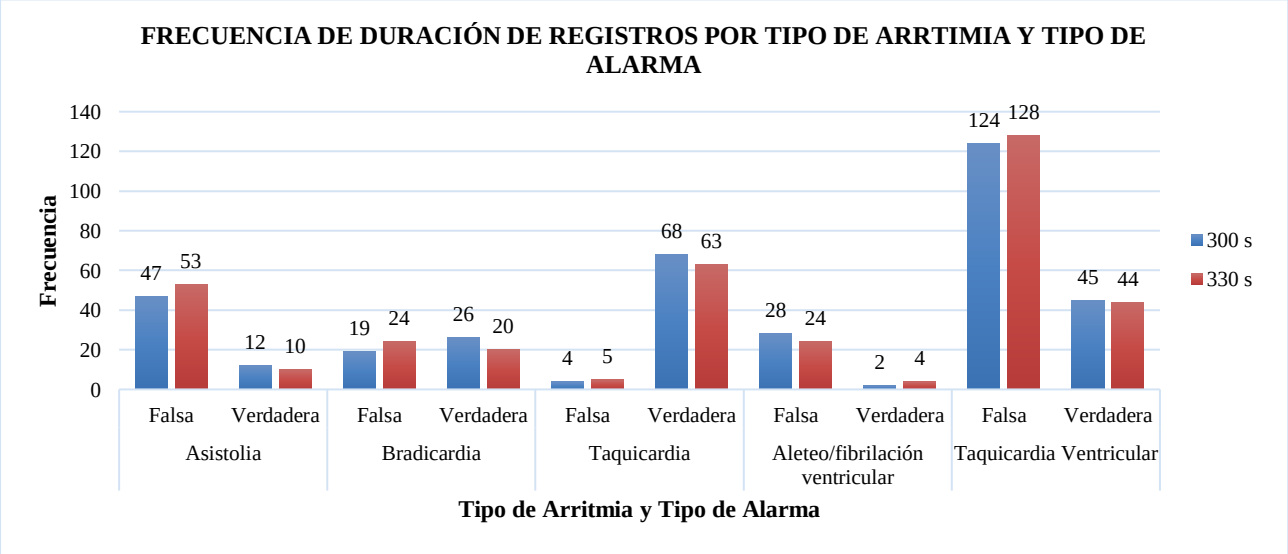

Fig. 3. Frecuencia de duración de registros por tipo de arrtimia y tipo de alarma

Los registros de señales fisiológicas se encuentran almacenados en archivos con extensión .mat y .hea, formatos binarios utilizados por MATLAB. Al realizar su procesamiento en Python, se encontró que el archivo .hea proporciona información detallada sobre los registros de señales fisiológicas, incluyendo datos críticos sobre la configuración del registro, las características de cada señal y los eventos clínicos detectados. Contiene detalles como el nombre del registro, que es único para cada dato; el número de señales, que usualmente es entre 3 y 4; la frecuencia de muestreo, que

es de 250 Hz; y el número total de muestras por señal, que varía entre 75.000 y 82.500. Cada señal contiene información sobre el formato, que es de 16+24 bits, la ganancia de la señal, la resolución de 16 bits por muestra, el desplazamiento, el valor inicial de la señal y el nombre de la señal. Además, el archivo incluye comentarios que describen eventos específicos detectados en la señal, como arritmias y si estos eventos fueron alarmas verdaderas o falsas.

En la siguiente tabla se puede ver de manera resumida la información proporcionada por este archivo:

TABLA III
DESCRIPCIÓN DE DATOS DEL ARCHIVO .HEA

Datos	Descripción
Nombre del registro	Identifica el archivo de señal fisiológica específico.
Número de señales	Indica cuántas señales fisiológicas se registraron (usualmente entre 3 y 4).
Frecuencia de muestreo	Frecuencia con la que se registran los datos en Hz (generalmente 250 Hz).
Número de muestras	Número total de muestras registradas en cada señal (varía entre 75000 y 82500).
Archivo de datos	Nombre del archivo que contiene los datos de la señal.
Formato de la señal	Detalles del formato de la señal, incluyendo bits por muestra y desplazamiento (16+24 bits).
Ganancia de la señal	Escala de medida de la señal, como x/mV para voltaje o y/mmHg para presión arterial.
Bits por muestra	Resolución en bits de cada muestra de la señal (16 bits).
Desplazamiento	Valor que se resta de las muestras de la señal.
Valor inicial de la señal	Valor de la primera muestra registrada.
Valor ajustado inicial	Valor inicial de la señal menos el desplazlamiento.
Nombre de la señal	Identificación de la señal específica, como II, V, ABP, RESP, PLETH.
Comentarios	Eventos clínicos detectados en la señal, como tipo de arritmia y si es una alarma falsa o verdadera.

Nota: Descripción detallada de los campos contenidos en el archivo .hea.

El archivo .mat proporciona los datos reales de las señales fisiológicas capturadas durante el monitoreo. Cada señal registrada en el archivo .hea se representa como una fila en una matriz contenida en el archivo .mat. Los valores en cada fila representan las mediciones de voltaje en diferentes puntos en el tiempo de monitoreo para cada señal específica. Es importante destacar que el archivo .mat puede incluir el valor de -32768 para indicar la ausencia de datos o puntos en los que la señal no fue registrada adecuadamente. Este valor, el más negativo que puede representar un entero de 16 bits (int16), actúa como un marcador distintivo que permite a los algoritmos de procesamiento identificar y manejar fácilmente estos segmentos de datos faltantes o corruptos durante el análisis.

TABLA IV
DESCRIPCIÓN DE DATOS DEL ARCHIVO .MAT

Fila	Señal
1	Canal II (ECG)
2	Canal V (ECG)
3	ABP (Presión Arterial)
4	RESP (Respiración)
5	PLETH (Plethysmography)

Nota: Listado de señales fisiológicas contenidas en el archivo .MAT.

4.2.2 Extracción y preparación de los datos del Canal II

Como parte del proceso de preparación de los datos, se seleccionó el Canal II debido a que proporciona una representación clara y consistente de las ondas P, QRS y T del ECG, lo que facilita la identificación de anomalías en el ritmo cardíaco [58].

El proceso de extracción se llevó a cabo emparejando los archivos con extensión .hea y .mat correspondientes a cada registro. Utilizando un script en Python y la biblioteca Scipy, se identificaron los segmentos relevantes de esta señal y se almacenaron en nuevos archivos con formato .npy para facilitar su procesamiento posterior.

Para cada registro, se realizaron los siguientes pasos:

I. Lectura de Archivos .hea y .mat: Se leyeron los archivos .hea para obtener los metadatos y los archivos .mat para acceder a los datos de las señales.

II. Identificación del Canal II: Se localizó y extrajo la señal correspondiente al Canal II de los datos contenidos en los archivos .mat.
III. Almacenamiento en Formato .npy: Las señales extraídas se guardaron en archivos .npy, un formato eficiente para el manejo de matrices en Python, junto con las etiquetas de las alarmas (falsas o verdaderas) y el tipo de arritmia presente.

De los 750 registros disponibles, se encontraron 10 que no contenían el Canal II. Como resultado, se cargaron un total de 740 señales y etiquetas correspondientes, listas para su procesamiento posterior.

TABLA V
FRECUENCIA DE ARRITMIAS EN REGISTROS SIN CANAL II

Tipo de Arritmia	Falsa Alarma	Alarma Verdadera
Bradicardia	2	1
Taquicardia	1	3
Taquicardia Ventricular	2	1
Total	5	5

Nota: Distribución de arritmias detectadas en registros que no incluyen el Canal II.

Esta metodología permitió estructurar y organizar las señales del ECG para su análisis y modelado. A continuación, se presenta una tabla resumida con la descripción de los datos extraídos:

TABLA VI
DESCRIPCIÓN DE DATOS EXTRAÍDOS Y ALMACENADOS

Campo	Descripción
Nombre del archivo .npy	Identificador único del archivo .npy generado para cada registro.
Señal extraída	Canal II.
Formato de almacenamiento	Formato .npy para datos y etiquetas.
Etiquetas	Tipo de alarma (falsa o verdadera) y tipo de arritmia (Asistolia, Bradicardia, Taquicardia, Aleteo/Fibrilación Ventricular, Taquicardia Ventricular).

Nota: Detalles de los datos extraídos del Canal II y almacenados en formato .npy, incluyendo etiquetas de alarmas y arritmias.

4.2.3 Preprocesamiento de las Señales

4.2.3.1 Eliminación de ruido en las señales

Las señales de ECG pueden contener varios tipos de ruido que afectan la precisión de los modelos de clasificación. Los principales tipos de ruido incluyen:

- Ruido de Línea Base: Fluctuaciones de baja frecuencia causadas por el movimiento del paciente o del equipo.
- Ruido de Alta Frecuencia: Interferencias de dispositivos electrónicos cercanos.
- Artefactos de Movimiento: Cambios abruptos en la señal debido al movimiento del paciente.
- Ruido Aleatorio: Variaciones impredecibles causadas por interferencias eléctricas, movimientos aleatorios del paciente y otros factores ambientales.

Para eliminar estos ruidos, se aplicaron los siguientes filtros y técnicas:

- Filtro de Paso Bajo: Para eliminar el ruido de alta frecuencia.
- Filtro de Paso Alto: Para eliminar el ruido de línea base.
- Filtro de Mediana: Para reducir los artefactos de movimiento.
- Filtro de Kalman: Para suavizar la señal y reducir el ruido aleatorio.

Estos pasos se implementaron secuencialmente para asegurar la limpieza y claridad de las señales.

4.2.3.2 Normalización Z-Score

Para garantizar una escala uniforme y mejorar el rendimiento del modelo, las señales fueron normalizadas utilizando la técnica de Z-Score. Este proceso ajusta las señales para que tengan una media de 0 y una desviación estándar de 1. Esto es crucial para evitar que las diferencias en escala afecten el comportamiento de los modelos.

4.2.3.3 Implementación del detector de Picos R

El detector de picos R es esencial para identificar los puntos característicos en un electrocardiograma (ECG) que representan la despolarización de los ventrículos del corazón [58]. Para esto, se utilizó un filtro Butterworth de orden 3 con frecuencias de corte específicas para

eliminar el ruido y resaltar los picos R. Los picos R se detectaron utilizando la función find_peaks de la biblioteca SciPy, ajustando parámetros como la altura mínima y la distancia mínima entre picos.

Para detectar los picos R en las señales de ECG, se implementó un algoritmo basado en el filtro de Pan y Tompkins. Este algoritmo incluye:

- Filtro de Paso Banda: Se aplica un filtro de paso banda para eliminar ruidos residuales y optimizar la señal específicamente para la detección de picos R.
- Derivada: Se calcula la derivada de la señal para resaltar los bordes de los picos R.
- Cuadrado de la Señal Derivada: La señal derivada se eleva al cuadrado para acentuar los picos y atenuar los valores pequeños, facilitando la detección de picos significativos.
- Promedio Móvil: Se utiliza un filtro de promedio móvil para suavizar la señal cuadrada, reduciendo aún más el ruido y facilitando la detección de los picos R.

4.2.3.4 Segmentación de las señales

Para facilitar la clasificación de los latidos, las señales se segmentaron en ventanas de tiempo basadas en los picos R detectados. Este enfoque se basa en estudios previos que han demostrado su eficacia en la segmentación de latidos de ECG [53]. Se utilizaron ventanas de 360 muestras (equivalentes a 1,44 segundos a una frecuencia de muestreo de 250 Hz) alrededor de cada pico R, asegurando que cada segmento contenga un complejo QRS completo.

Resultados de la segmentación son:

- Señales segmentadas y filtradas: 851.309
- Etiquetas correspondientes: 851.309

4.2.3.5 Preparación de datos para el modelo

Finalmente, las señales segmentadas se organizaron en conjuntos de entrenamiento y prueba. Las etiquetas correspondientes, que incluyen el tipo de alarma (falsa o verdadera) y el tipo de arritmia (Asistolia, Bradicardia, Taquicardia, Aleteo/Fibrilación Ventricular, Taquicardia Ventricular), también se prepararon para el entrenamiento de los modelos. Los datos segmentados se dividieron en conjuntos de entrenamiento y prueba en una proporción de 80% y 20%,

respectivamente. Se estructuraron de la siguiente manera:

TABLA
VII DESCRIPCIÓN DE LOS DATOS SEGMENTADOS Y ETIQUETADOS

Tipo de Datos	Entrenamiento	Prueba
Señales de ECG	681.047 registros	170.262 registros
Longitud de Señal (muestras)	360	360
Canales de Señal	1	1
Etiquetas de Alarma	681.047 etiquetas	170.262 etiquetas
Dimensiones de Alarma	2 (falsa/verdadera)	2 (falsa/verdadera)
Etiquetas de Arritmia	681.047 etiquetas	170.262 etiquetas
Dimensiones de Arritmia	5 tipos	5 tipos

Nota: Características de los datos de ECG segmentados y etiquetados para los conjuntos de entrenamiento y prueba, incluyendo el número de registros, longitud de la señal, canales de señal, y las etiquetas correspondientes a alarmas y arritmias.

4.3 IMPLEMENTACIÓN DE MODELOS

4.3.1 Primer modelo: Clasificación con Red Neuronal Convolucional

4.3.1.1 Arquitectura del Modelo

En la implementación del modelo, se diseñó y entrenó una red neuronal convolucional (CNN) con el objetivo de clasificar alarmas y tipos de arritmias a partir de señales de ECG preprocesadas. El modelo se construyó utilizando las librerías TensorFlow y Keras, y constó de un total de 14 capas, incluyendo capas convolucionales, de max-pooling, de aplanado, densas y de dropout.

A continuación, se describen en detalle las capas que componen la arquitectura del modelo:

- Entrada: La capa de entrada recibe señales ECG preprocesadas con una dimensión de entrada de 360 muestras de un solo canal (360, 1).
- Capas Convolucionales:
  - Primera Capa Convolucional: Se utilizaron 64 filtros con un tamaño de kernel de 3 y función de activación ReLU, seguida de una capa de max-pooling con un tamaño de pool de 2. Esta capa extrae características locales de la señal de ECG.
  - Segunda y Tercera Capas Convolucionales: Ambas capas también aplican 64 filtros con un tamaño de kernel de 3 y función de activación ReLU, seguidas de capas de max-pooling con un tamaño de pool de 2. Estas capas profundizan la extracción de

características relevantes de la señal.

- Capa de Flatten: Convierte la salida de las capas convolucionales en un vector unidimensional, preparando los datos para las capas densas.
- Capas Densas:
  - Primera Capa Densa: Contiene 128 neuronas con activación ReLU, seguida de una capa de dropout con una tasa del 50% para prevenir el sobreajuste.
  - Segunda Capa Densa: Contiene 64 neuronas con activación ReLU, seguida de otra capa de dropout con una tasa del 50%.
- Salidas del Modelo:
  - Salida de Alarma: Una capa densa con 2 neuronas y activación softmax para la clasificación binaria de alarmas.
  - Salida de Arritmia: Una capa densa con 5 neuronas y activación softmax para la clasificación de tipos de arritmias.

TABLA VIII
ARQUITECTURA DEL MODELO USANDO RED NEURONAL CONVOLUCIONAL

Nº	Capa	Tipo	Forma de Salida	Parámetros
1	Entrada	Input	(360, 1)	0
2	Convolucional 1	Conv1D	(358, 64)	256
3	Max Pooling 1	MaxPooling1D	(179, 64)	0
4	Convolucional 2	Conv1D	(177, 64)	12.352
5	Max Pooling 2	MaxPooling1D	(88, 64)	0
6	Convolucional 3	Conv1D	(86, 64)	12.352
7	Max Pooling 3	MaxPooling1D	(43, 64)	0
8	Aplanado	Flatten	(2752)	0
9	Densa 1	Dense	(128)	352.384
10	Dropout 1	Dropout	(128)	0
11	Densa 2	Dense	(64)	8.256
12	Dropout 2	Dropout	(64)	0
13	Salida de Alarma	Dense	(2)	130
14	Salida de Arritmia	Dense	(5)	325
	Total			386.055

Nota: Detalles de la arquitectura del modelo basado en una red neuronal convolucional (CNN), incluyendo las capas, tipo, forma de salida y el número de parámetros por capa.

El modelo CNN se entrenó utilizando el optimizador Adam y la función de pérdida de entropía cruzada categórica para ambas salidas. El proceso de entrenamiento se llevó a cabo durante 20 épocas con un tamaño de batch de 32. Se utilizó un conjunto de datos de entrenamiento y un conjunto de validación para monitorear el rendimiento del modelo.

4.3.1.2 Ajuste de Hiperparámetros

Para optimizar el rendimiento del modelo CNN, se realizó un ajuste de hiperparámetros mediante la técnica de búsqueda aleatoria utilizando la biblioteca Keras Tuner. Los hiperparámetros ajustados incluyeron el número de filtros en las capas convolucionales, el número de unidades en las capas densas y la tasa de dropout.

Los hiperparámetros ajustados fueron:

- Filtros de la primera capa convolucional: Rango entre 32 y 128.
- Filtros de la segunda y tercera capas convolucionales: Rango entre 32 y 128.
- Unidades de la capa densa: Rango entre 64 y 256.
- Tasa de dropout: Rango entre 0,2 y 0,5.

Después de ajustar los hiperparámetros, un nuevo modelo se entrenó desde cero utilizando los mejores hiperparámetros encontrados. Este modelo ajustado también se entrenó durante 20 épocas con un tamaño de batch de 32, utilizando el mismo conjunto de datos de entrenamiento y validación para asegurar una comparación justa con el modelo incial.

4.3.2 Segundo modelo: Clasificación con Red Neuronal Recurrente

4.3.2.1 Arquitectura del Modelo

En la implementación del modelo, se diseñó y entrenó una red neuronal recurrente (RNN) con capas LSTM (Long Short-Term Memory) para clasificar alarmas y tipos de arritmias a partir de señales de ECG preprocesadas. El modelo se construyó utilizando las librerías TensorFlow y Keras, y constó de un total de 14 capas, incluyendo capas LSTM, de dropout y densas.

El modelo RNN inicial se diseñó con la siguiente arquitectura:

- Entrada: La capa de entrada recibe señales ECG preprocesadas con una dimensión de entrada de 360 muestras de un solo canal (360, 1).

- Capas LSTM:
  - Primera Capa LSTM: 64 unidades con retorno de secuencias, seguida de una capa de dropout con una tasa del 50%.
  - Segunda Capa LSTM: 64 unidades con retorno de secuencias, seguida de una capa de dropout con una tasa del 50%.
  - Tercera Capa LSTM: 64 unidades sin retorno de secuencias, seguida de una capa de dropout con una tasa del 50%.
- Capa de Flatten: Convierte la salida de las capas LSTM en un vector unidimensional, preparando los datos para las capas densas.
- Capas Densas:
  - Primera Capa Densa: Contiene 128 neuronas con activación ReLU, seguida de una capa de dropout con una tasa del 50%.
  - Segunda Capa Densa: Contiene 64 neuronas con activación ReLU, seguida de otra capa de dropout con una tasa del 50%.
- Salidas del Modelo:
  - Salida de Alarma: Una capa densa con 2 neuronas y activación softmax para la clasificación binaria de alarmas.
  - Salida de Arritmia: Una capa densa con 5 neuronas y activación softmax para la clasificación de tipos de arritmias.

El modelo RNN se entrenó utilizando el optimizador Adam y la función de pérdida de entropía cruzada categórica para ambas salidas. El proceso de entrenamiento se llevó a cabo durante 20 épocas con un tamaño de batch de 32. Se utilizó un conjunto de datos de entrenamiento y un conjunto de validación para monitorear el rendimiento del modelo. Las librerías utilizadas para el entrenamiento incluyen TensorFlow y Keras.

TABLA IX
ARQUITECTURA DEL MODELO USANDO RED NEURONAL RECURRENTE

Nº	Capa	Tipo	Forma de Salida	Parámetros
1	Entrada	Input	(360, 1)	0
2	LSTM 1	LSTM	(360, 64)	16.896
3	Dropout 1	Dropout	(360, 64)	0
4	LSTM 2	LSTM	(360, 64)	33.024
5	Dropout 2	Dropout	(360, 64)	0
6	LSTM 3	LSTM	(64)	33.024
7	Dropout 3	Dropout	(64)	0
8	Aplanado	Flatten	(64)	0
9	Densa 1	Dense	(128)	8.320
10	Dropout 4	Dropout	(128)	0
11	Densa 2	Dense	(64)	8.256
12	Dropout 5	Dropout	(64)	0
13	Salida de Alarma	Dense	(2)	130
14	Salida de Arritmia	Dense	(5)	325
	Total			99.975

Nota: Detalles de la arquitectura del modelo basado en una red neuronal recurrente (RNN) utilizando capas LSTM, incluyendo las capas, tipo, forma de salida y el número de parámetros por capa.

4.3.2.2 Ajuste de Hiperparámetros

Para optimizar el rendimiento del modelo RNN, se realizó un ajuste de hiperparámetros mediante la técnica de búsqueda aleatoria utilizando la biblioteca Keras Tuner. Los hiperparámetros ajustados incluyeron el número de unidades en las capas LSTM, el número de unidades en las capas densas y la tasa de dropout.

Los hiperparámetros ajustados fueron:

- Unidades en la primera capa LSTM: Rango entre 32 y 256.
- Unidades en la segunda y tercera capas LSTM: Rango entre 32 y 256.
- Unidades en la capa densa: Rango entre 64 y 256.
- Tasa de dropout: Rango entre 0,2 y 0,5.

Después de ajustar los hiperparámetros, un nuevo modelo se entrenó desde cero utilizando los mejores hiperparámetros encontrados. Este modelo ajustado también se entrenó durante 20 épocas con un tamaño de batch de 32, utilizando el mismo conjunto de datos de entrenamiento y validación para asegurar una comparación justa entre los dos enfoques.

4.4 EVALUACIÓN

4.4.1 Evaluación del Modelo CNN

4.4.1.1 Evaluación del Modelo CNN Inicial

El modelo CNN inicial se evaluó utilizando un conjunto de prueba, obteniendo las siguientes métricas de rendimiento:

- Precisión de alarmas: 95,34%
- Precisión de arritmias: 88,17%

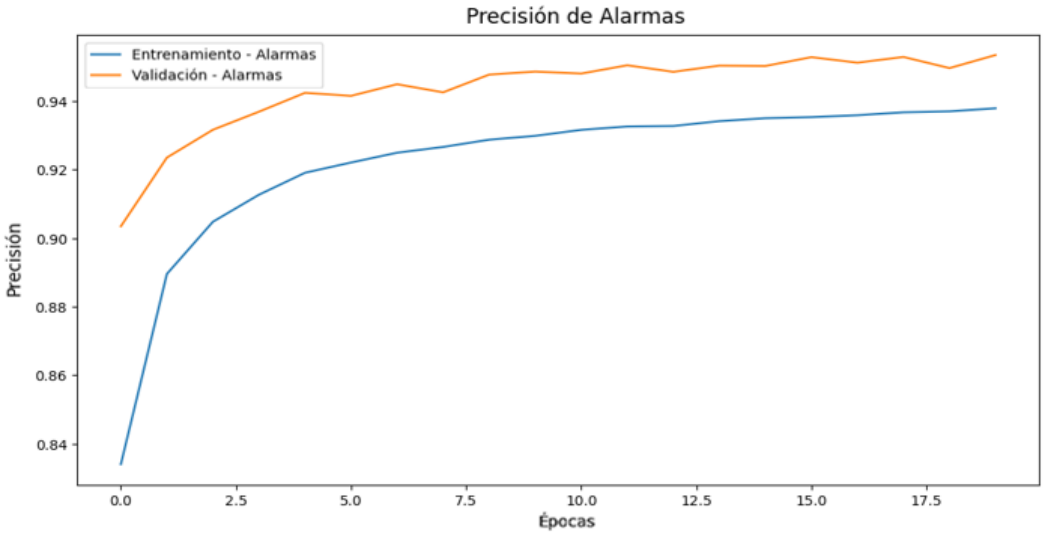

Fig. 4 Precisión de Entrenamiento y Validación del Modelo CNN Inicial para Alarmas
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de alarmas en el modelo CNN inicial a lo largo de 20 épocas. Se observa una mejora constante en la precisión.

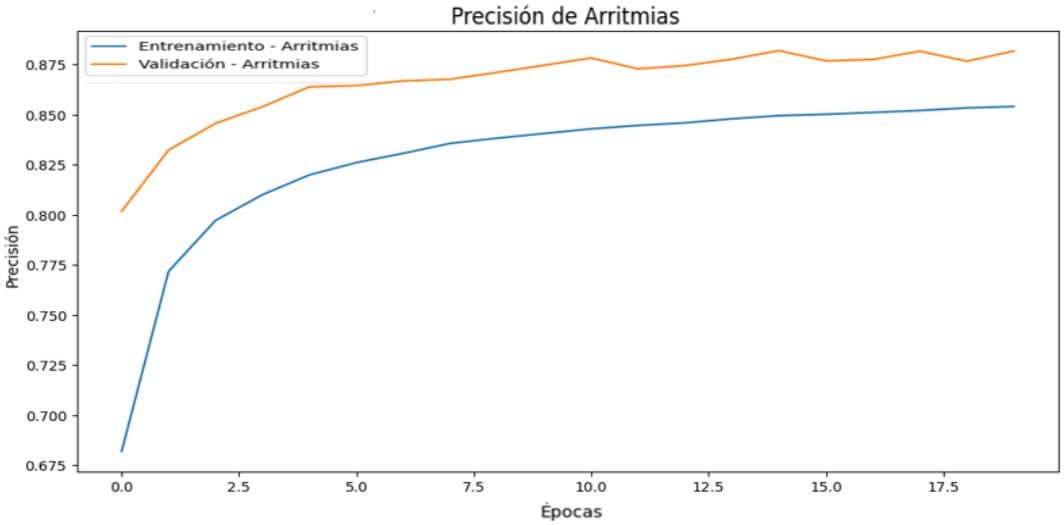

Fig. 5 Precisión de Entrenamiento y Validación del Modelo CNN Inicial para Arritmias
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de arritmias en el modelo CNN inicial a lo largo de 20 épocas. Se observa una mejora significativa en la precisión.

A continuación, se detallan las métricas de precisión, sensibilidad y F1-Score para cada clase de alarma y arritmia.

TABLA X
EVALUACIÓN DE CLASIFICACIÓN PARA ALARMAS MODELO CNN

Clase	Precisión	Sensibilidad	F1-Score
Falsa alarma	0,94	0,98	0,96
Verdadera alarma	0,97	0,91	0,94

Nota: Resultados de la evaluación de clasificación del modelo CNN para alarmas, mostrando precisión, sensibilidad y F1-Score para clases de falsa alarma y verdadera alarma.

TABLA XI
EVALUACIÓN DE CLASIFICACIÓN PARA ARRITMIAS MODELO CNN

Clase	Precisión	Sensibilidad	F1-Score
Asistolia	0,91	0,79	0,84
Bradicardia	0,97	0,78	0,87
Taquicardia	0,95	0,93	0,94
Aleteo/Fibrilación Ventricular	0,90	0,62	0,73
Taquicardia Ventricular	0,83	0,96	0,89

Nota: Resultados de la evaluación de clasificación del modelo CNN para diferentes tipos de arritmias, mostrando precisión, sensibilidad y F1-Score para cada clase de arritmia.

El modelo inicial tiene una alta precisión y sensibilidad en la clasificación de alarmas y tipos de arritmias, con una notable precisión en la detección de taquicardia y bradicardia.

4.4.1.2 Evaluación del Modelo CNN Ajustado

Utilizando la técnica de búsqueda aleatoria con Keras Tuner, se realizaron 10 iteraciones para encontrar los mejores hiperparámetros que resultaron en mejoras significativas en las métricas de precisión de validación.

Los mejores hiperparámetros fueron:

- Filtros de la primera capa convolucional: 96
- Filtros de la segunda capa convolucional: 128
- Filtros de la tercera capa convolucional: 96
- Unidades de la capa densa: 256
- Tasa de dropout: 0,2

El modelo inicial, antes del ajuste de hiperparámetros, tenía un total de 386.055 parámetros. Tras el ajuste de hiperparámetros, el modelo contaba con 1.133.159 parámetros, lo que representa un aumento significativo en la capacidad del modelo para aprender características relevantes de las señales ECG.

TABLA XII
HIPERPARÁMETROS AJUSTADOS PARA EL MODELO CON RED NEURONAL CONVOLUCIONAL

Hiperparámetro	Valor Inicial	Rango de Búsqueda	Mejor Valor
Filtros Convolucional 1	64	[32, 64, 96, 128]	96
Filtros Convolucional 2	64	[32, 64, 96, 128]	128
Filtros Convolucional 3	64	[32, 64, 96, 128]	96
Unidades Capa Densa	128	[64, 128, 192, 256]	256
Tasa de Dropout	0,5	[0.2, 0.3, 0.4, 0.5]	0,2
Total Parámetros Inicial	386.055		
Total Parámetros Ajustado	N/A		1.133.159

Nota: Valores iniciales, rango de búsqueda y mejores valores encontrados para los hiperparámetros ajustados en el modelo con red neuronal convolucional (CNN).

Las métricas obtenidas después de ajustar los hiperparámetros fueron las siguientes:

- Precisión de alarmas: 96,69%
- Precisión de arritmias: 91,17%

Estos resultados demuestran que el ajuste de hiperparámetros ha sido efectivo, mejorando significativamente la precisión del modelo en la detección de alarmas y la clasificación de arritmias en las señales ECG. El modelo ajustado muestra una mayor capacidad para aprender y generalizar características relevantes, lo que es crucial para aplicaciones prácticas en el ámbito médico.

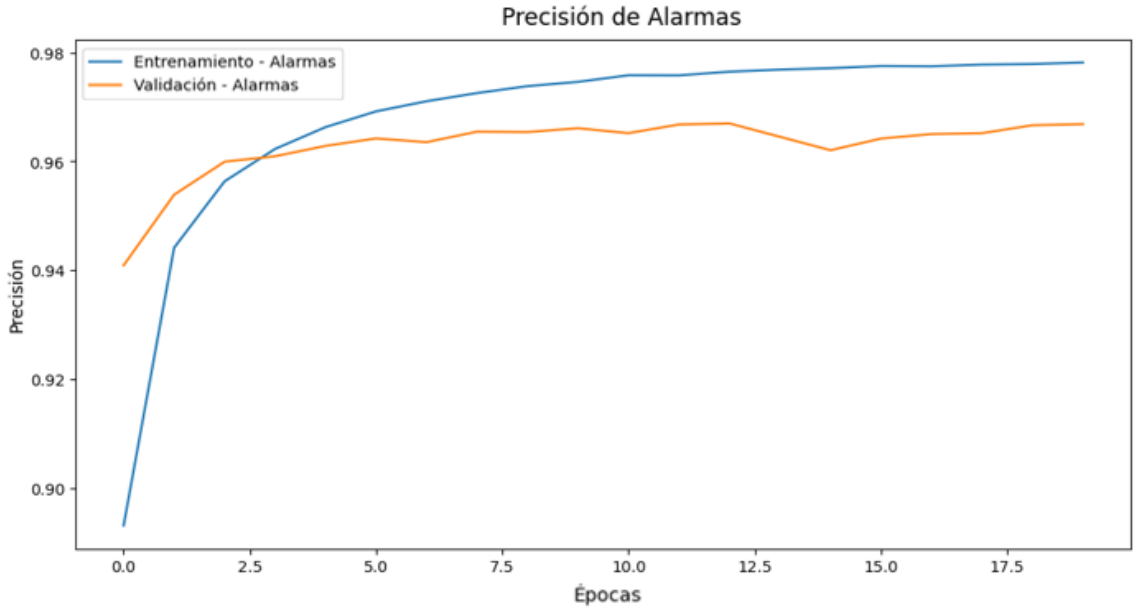

Fig. 6 Precisión de Entrenamiento y Validación del Modelo CNN Ajustado para Alarmas
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de alarmas en el modelo CNN ajustado a lo largo de 20 épocas. Se observa una mejora significativa en la precisión.

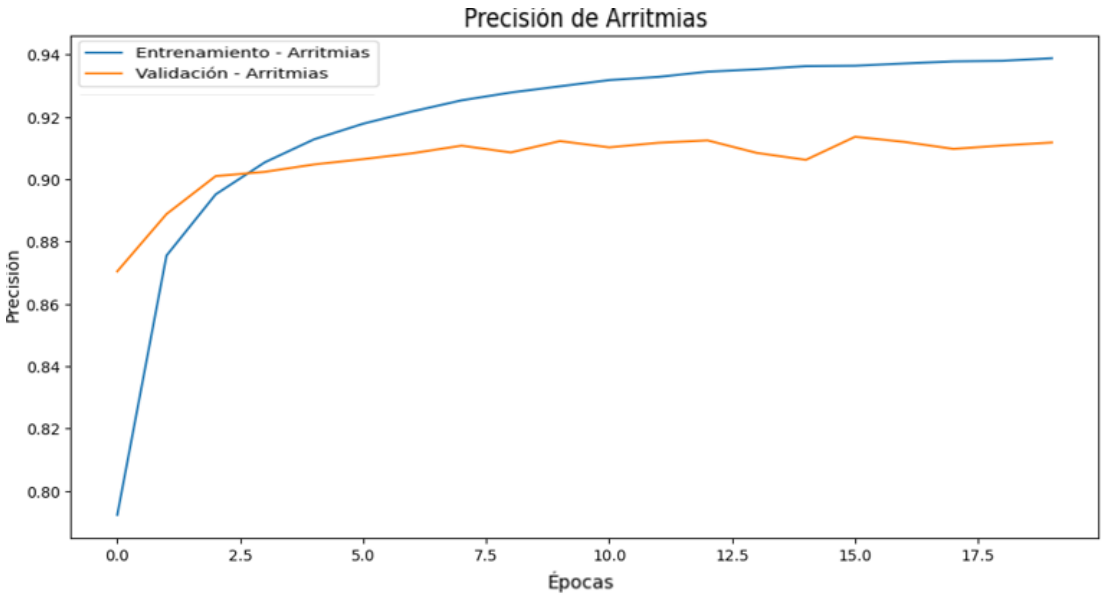

Fig. 7 Precisión de Entrenamiento y Validación del Modelo CNN Ajustado para Arritmias

Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de arritmias en el modelo CNN ajustado a lo largo de 20 épocas. Se observa una mejora notable en la precisión.

A continuación, se detallan las métricas de precisión, sensibilidad y F1-Score para cada clase de alarma y arritmia después del ajuste de hiperparámetros.

TABLA XIII
EVALUACIÓN DE CLASIFICACIÓN PARA ALARMAS MODELO CNN AJUSTADO

Clase	Precisión	Sensibilidad	F1-Score
Falsa alarma	0,96	0,98	0,97
Verdadera alarma	0,97	0,94	0,96

Nota: Resultados de la evaluación de clasificación del modelo CNN ajustado para alarmas, mostrando precisión, sensibilidad y F1-Score para las clases de falsa alarma y verdadera alarma.

TABLA XIV
EVALUACIÓN DE CLASIFICACIÓN PARA ARRITMIAS MODELO CNN AJUSTADO

Clase	Precisión	Sensibilidad	F1-Score
Asistolia	0,9	0,87	0,88
Bradicardia	0,92	0,87	0,9
Taquicardia	0,97	0,95	0,96
Aleteo/Fibrilación Ventricular	0,83	0,77	0,8
Taquicardia Ventricular	0,9	0,94	0,92

Nota: Resultados de la evaluación de clasificación del modelo CNN ajustado para diferentes tipos de arritmias, mostrando precisión, sensibilidad y F1-Score para cada clase de arritmia.

El modelo ajustado mostró mejoras en todas las métricas de rendimiento, destacándose una mayor precisión en la clasificación de alarmas y arritmias.

4.4.2 Evaluación del Modelo RNN

4.4.2.1 Evaluación del Modelo RNN Inicial

El modelo RNN inicial se evaluó utilizando un conjunto de prueba, obteniendo las siguientes métricas de rendimiento:

- Precisión de alarmas: 96,94%
- Precisión de arritmias: 91,25%

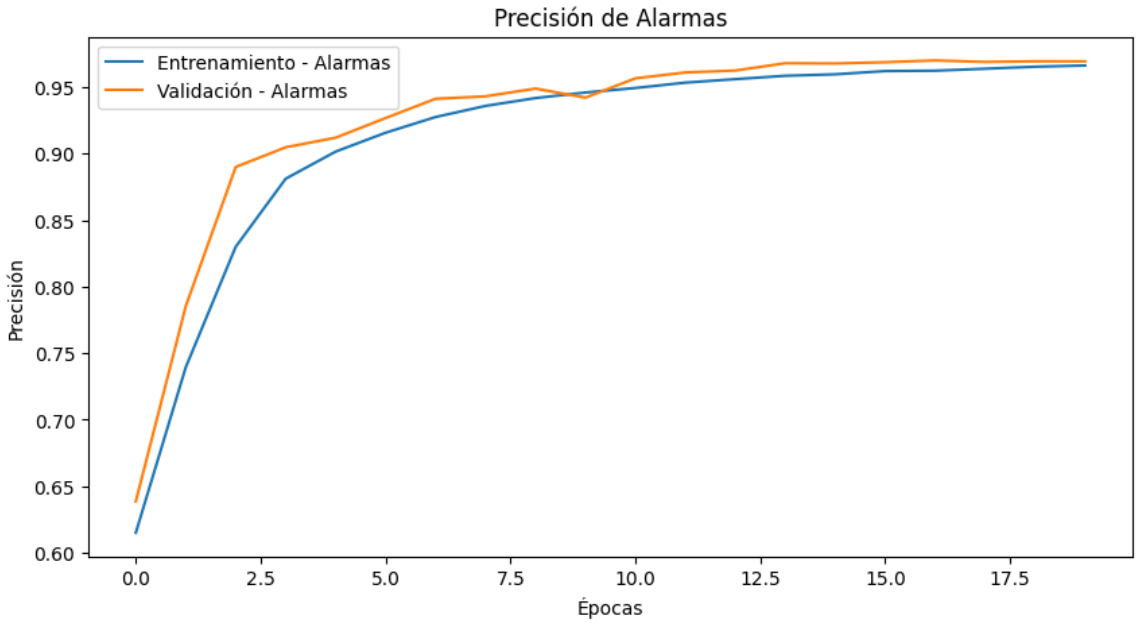

Fig. 8 Precisión de Entrenamiento y Validación del Modelo RNN Inicial para Alarmas
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de alarmas en el modelo RNN inicial a lo largo de 20 épocas. Se observa una mejora constante en la precisión.

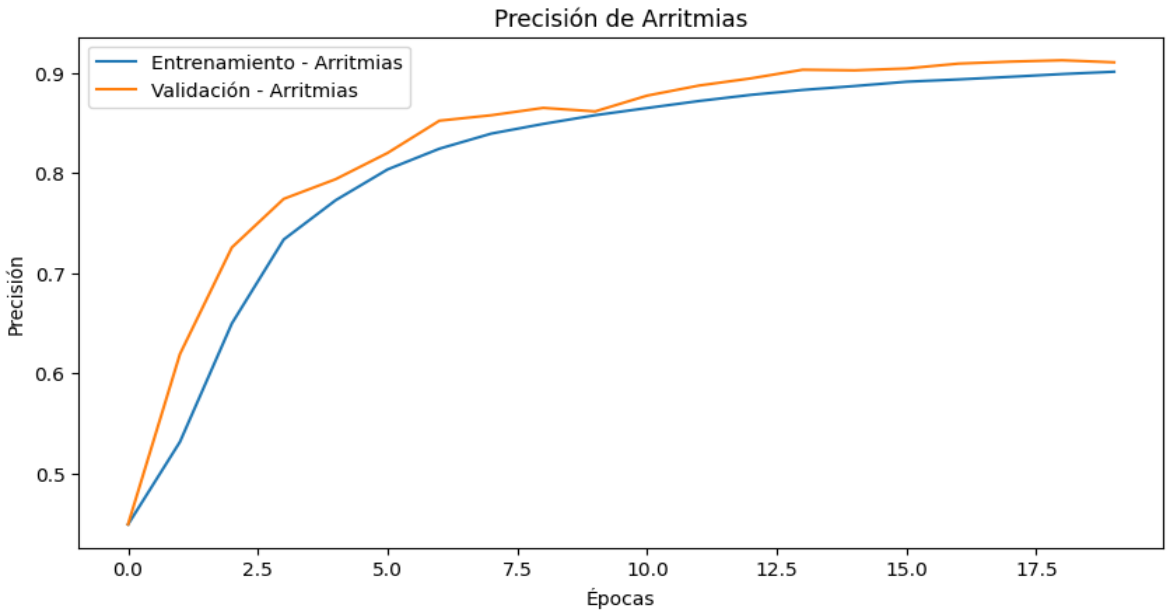

Fig. 9 Precisión de Entrenamiento y Validación del Modelo RNN Inicial para Arritmias

Descripción: Esta gráfica presenta la evolución de la precisión de entrenamiento y validación para la clasificación de arritmias en el modelo RNN inicial a lo largo de 20 épocas.

A continuación, se detallan las métricas de precisión, sensibilidad y F1-Score para cada clase de alarma y arritmia.

TABLA XV
EVALUACIÓN DE CLASIFICACIÓN PARA ALARMAS MODELO RNN

Clase	Precisión	Sensibilidad	F1-Score
Falsa alarma	0,98	0,97	0,98
Verdadera alarma	0,96	0,96	0,96

Nota: Resultados de la evaluación de clasificación del modelo RNN para alarmas, mostrando precisión, sensibilidad y F1-Score para las clases de falsa alarma y verdadera alarma.

TABLA XVI
EVALUACIÓN DE CLASIFICACIÓN PARA ARRITMIAS MODELO RNN

Clase	Precisión	Sensibilidad	F1-Score
Asistolia	0,91	0,85	0,88
Bradicardia	0,94	0,86	0,90
Taquicardia	0,96	0,97	0,96
Aleteo/Fibrilación Ventricular	0,85	0,77	0,81
Taquicardia Ventricular	0,90	0,95	0,92

Nota: Resultados de la evaluación de clasificación del modelo RNN para diferentes tipos de arritmias, mostrando precisión, sensibilidad y F1-Score para cada clase de arritmia.

El modelo inicial usando RNN mostró un buen rendimiento en la clasificación de alarmas y arritmias, especialmente en la detección de bradicardia y taquicardia.

4.4.2.2 Resultados del Modelo RNN Ajustado

Tras de 5 iteraciones con la técnica de búsqueda aleatoria de Keras Tuner, se identificaron los mejores hiperparámetros, que se detallan a continuación:

- Unidades en la primera capa LSTM: 32
- Unidades en la segunda capa LSTM: 32
- Unidades en la tercera capa LSTM: 96
- Unidades en la primera capa densa: 128
- Unidades en la segunda capa densa: 64
- Tasa de dropout en la primera capa LSTM: 0.5
- Tasa de dropout en la segunda capa LSTM: 0.2
- Tasa de dropout en la tercera capa LSTM: 0.4
- Tasa de dropout en la primera capa densa: 0.3
- Tasa de dropout en la segunda capa densa: 0.1

Se construyó y entrenó un nuevo modelo desde cero utilizando estos hiperparámetros. El modelo inicial tenía un total de 99.975 parámetros, mientras que el modelo ajustado contaba con 83.335 parámetros.

Se puede observar que el ajuste de hiperparámetros ha permitido una reducción en el número total de parámetros del modelo, lo cual contribuye a una mayor eficiencia en el entrenamiento y la predicción sin sacrificar la precisión del modelo.

TABLA XVII
HIPERPARÁMETROS AJUSTADOS PARA EL MODELO CON RED NEURONAL RECURRENTE

Hiperparámetro	Valor Inicial	Rango de Búsqueda	Mejor Valor
Unidades LSTM 1	64	[32, 64, 128]	32
Tasa de Dropout 1	0,5	[0.1, 0.2, 0.3, 0.4, 0.5]	0,5
Unidades LSTM 2	64	[32, 64, 128]	32
Tasa de Dropout 2	0,5	[0.1, 0.2, 0.3, 0.4, 0.5]	0,2
Unidades LSTM 3	64	[32, 64, 128]	96
Tasa de Dropout 3	0,5	[0.1, 0.2, 0.3, 0.4, 0.5]	0,4
Unidades Densa 1	128	[64, 128, 256]	128
Tasa de Dropout 4	0,5	[0.1, 0.2, 0.3, 0.4, 0.5]	0,3
Unidades Densa 2	64	[32, 64, 96, 128]	64
Tasa de Dropout 5	0,5	[0.1, 0.2, 0.3, 0.4, 0.5]	0,1
Total Parámetros Inicial	99.975		
Total Parámetros Ajustado	N/A		83.335

Nota: Valores iniciales, rango de búsqueda y mejores valores encontrados para los hiperparámetros ajustados en el modelo con red neuronal recurrente (RNN).

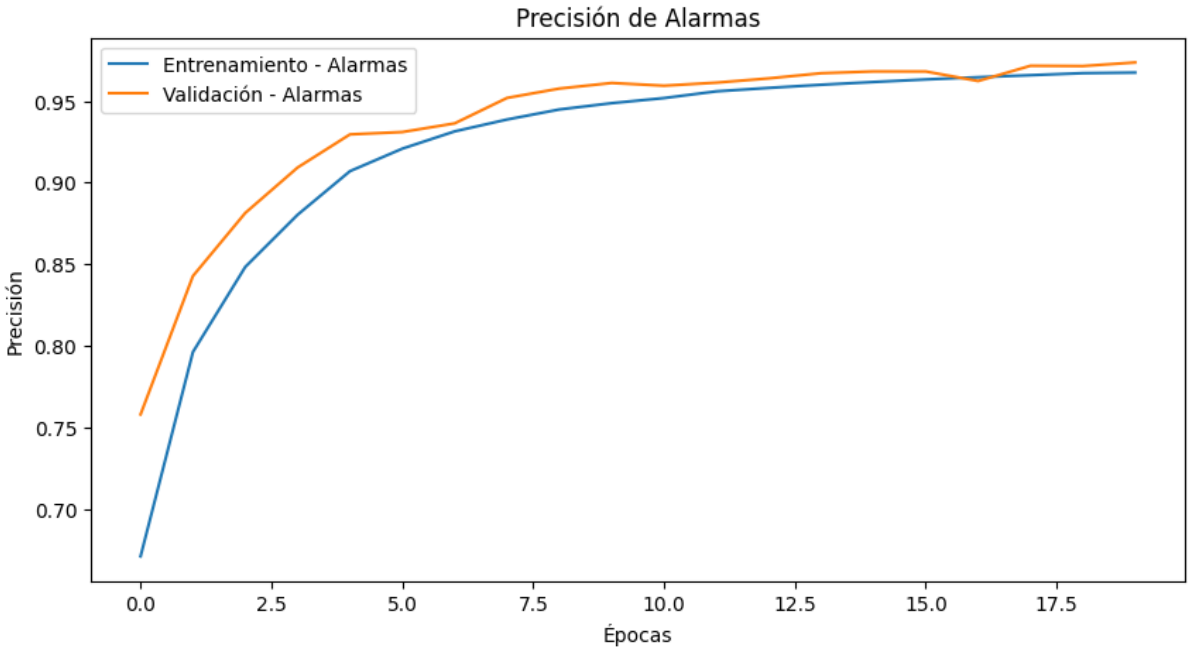

Fig. 10 Precisión de Entrenamiento y Validación del Modelo RNN ajustado para Alarmas
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de alarmas en el modelo CNN ajustado a lo largo de 20 épocas. Se observa una mejora notable en la precisión.

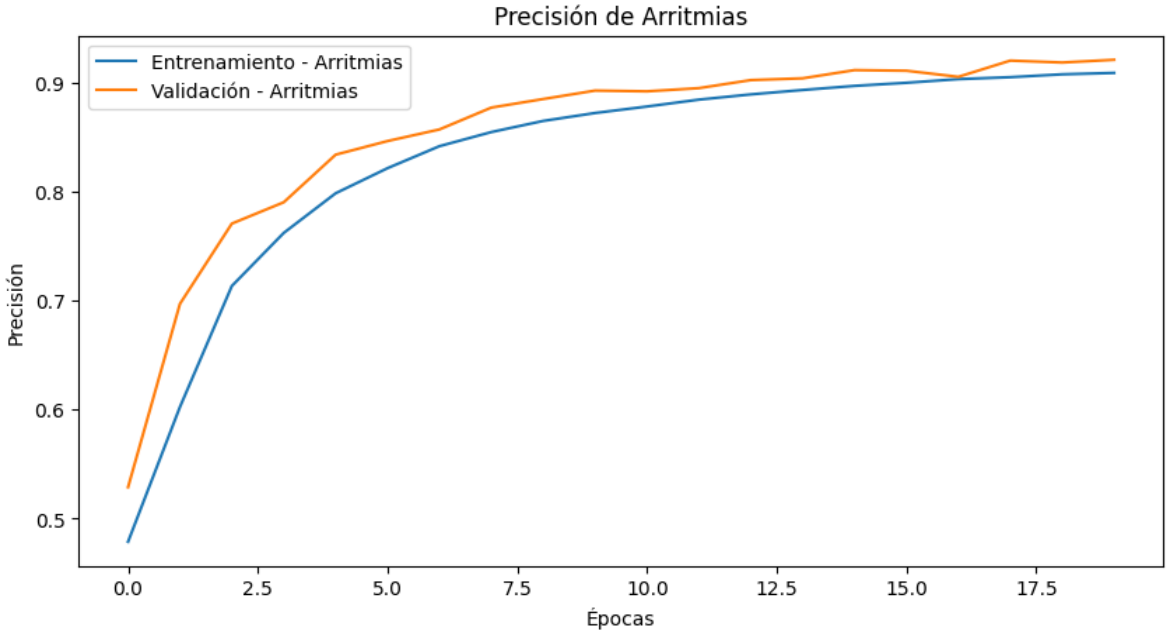

Fig. 11 Precisión de Entrenamiento y Validación del Modelo RNN ajustado para Arritmias
Descripción: Esta gráfica muestra la evolución de la precisión de entrenamiento y validación para la clasificación de arritmias en el modelo RNN ajustado a lo largo de 20 épocas. Se observa una mejora significativa en la precisión.

Las métricas obtenidas tras el ajuste de hiperparámetros fueron las siguientes:

- Precisión de alarmas: 97,36%
- Precisión de arritmias: 92,10%

A continuación, se detallan las métricas de precisión, sensibilidad y F1-Score para cada clase de alarma y arritmia después del ajuste de hiperparámetros.

TABLA XVIII
EVALUACIÓN DE CLASIFICACIÓN PARA ALARMAS MODELO RNN AJUSTADO

Clase	Precisión	Sensibilidad	F1-Score
Falsa alarma	0,97	0,98	0,98
Verdadera alarma	0,98	0,96	0,97

Nota: Resultados de la evaluación de clasificación del modelo RNN ajustado para alarmas, mostrando precisión, sensibilidad y F1-Score para las clases de falsa alarma y verdadera alarma.

TABLA XIX
EVALUACIÓN DE CLASIFICACIÓN PARA ARRITMIAS MODELO RNN AJUSTADO

Clase	Precisión	Sensibilidad	F1-Score
Asistolia	0,90	0,88	0,89
Bradicardia	0,95	0,85	0,90
Taquicardia	0,97	0,97	0,97
Aleteo/Fibrilación Ventricular	0,87	0,76	0,82
Taquicardia Ventricular	0,91	0,96	0,93

Nota: Resultados de la evaluación de clasificación del modelo RNN ajustado para diferentes tipos de arritmias, mostrando precisión, sensibilidad y F1-Score para cada clase de arritmia.

4.4.3 Comparación de Modelos

Para determinar el mejor modelo, se compararon las métricas de evaluación clave:

TABLA XX
COMPARACIÓN MÉTRICAS DE RENDIMIENTO DE LOS MODELOS

Métrica	CNN Inicial	CNN Ajustado	RNN Inicial	RNN Ajustado
Precisión Alarma	95,34%	96,69%	96,94%	97,36%
Precisión Arritmia	88,17%	91,17%	91,25%	92,10%
Sensibilidad Alarma	94,50%	96%	87%	96%
Sensibilidad Arritmia	81,60%	88%	67%	85%
F1-Score Alarma	95%	97%	88%	97%
F1-Score Arritmia	88%	91%	73%	89%

Nota: Comparación de las métricas de rendimiento entre los modelos CNN y RNN, tanto en su versión inicial como ajustada, incluyendo precisión, sensibilidad y F1-Score para la clasificación de alarmas y arritmias.

Esta tabla compara las métricas de rendimiento de los modelos CNN y RNN antes y después del ajuste de hiperparámetros. Se observa una mejora significativa en las métricas de rendimiento de los modelos ajustados en comparación con los iniciales, destacando la eficacia del ajuste de hiperparámetros en mejorar la precisión, sensibilidad y F1-Score de los modelos.

Por lo tanto, se selecciona el modelo RNN ajustado para la implementación de una interfaz que permita visualizar cómo se detectan y clasifican las arritmias. Este modelo proporciona una combinación robusta de alta precisión y sensibilidad, lo cual es crucial para aplicaciones clínicas donde la detección precisa de arritmias es esencial para la atención y el diagnóstico oportuno de los pacientes.

4.5 INTERFAZ WEB

La implementación de interfaces intuitivas no solo facilita el acceso a herramientas avanzadas, sino que también promueve su uso generalizado, permitiendo que profesionales sin conocimientos técnicos profundos puedan beneficiarse de las innovaciones tecnológicas. Una interfaz de usuario bien diseñada aumenta la probabilidad de adopción de nuevas tecnologías y herramientas, dado que los usuarios pueden aprender y utilizar las aplicaciones más rápidamente y con menos errores [59]. Esta afirmación subraya la importancia de desarrollar aplicaciones con interfaces accesibles para asegurar que los avances en tecnología, como los modelos de aprendizaje profundo para la detección de arritmias, puedan ser utilizados efectivamente en entornos clínicos, académicos y de investigación.

Por esta razón, se diseñó una interfaz web que sirve como herramienta para la detección y clasificación de arritmias, facilitando su uso por parte de profesionales de la salud y técnicos en diferentes lugares. La interfaz amigable y los procesos automatizados aseguran que el sistema pueda ser utilizado sin necesidad de conocimientos técnicos profundos.

4.5.1 Descripción de la implementación

Para facilitar el uso del modelo de aprendizaje profundo desarrollado para la detección y clasificación de arritmias, se creó una interfaz web utilizando la biblioteca Streamlit. Esta herramienta permite la creación rápida y sencilla de aplicaciones web interactivas en Python. La implementación se realizó siguiendo estos pasos:

4.5.1.1 Instalación de dependencias

Se identificaron y se instalaron las bibliotecas necesarias para el desarrollo de la aplicación. Streamlit se utilizó para la creación de la interfaz web debido a su facilidad de uso y capacidad para generar aplicaciones interactivas. Pyngrok se empleó para crear túneles de red, permitiendo el acceso remoto seguro a la aplicación. Además, se instalaron bibliotecas especializadas como h5py, wfdb y scipy, que son esenciales para el manejo y procesamiento de archivos de señales fisiológicas. Estas bibliotecas permiten la lectura y manipulación de los formatos de archivos utilizados en el proyecto (.hea y .mat), así como la implementación de técnicas avanzadas de procesamiento de señales.

4.5.1.2 Configuración de Ngrok

Ngrok se configuró para facilitar el acceso a la aplicación desde cualquier ubicación sin necesidad de configuraciones complicadas de red. Ngrok crea un túnel seguro desde una URL pública hacia la aplicación local, lo que es especialmente útil para demostraciones y pruebas en entornos donde la configuración de red puede ser una barrera. Se estableció un token de autenticación para asegurar la conexión y se aseguraron que todos los túneles preexistentes estuvieran cerrados para evitar conflictos de conexión.

4.5.1.3 Creación del archivo de la aplicación

Se desarrolló un archivo Python (app.py) que contiene la lógica completa de la aplicación. Este archivo define la estructura de la interfaz de usuario, las funciones de procesamiento de señales y las rutinas de predicción. La interfaz se diseñó para ser intuitiva, guiando al usuario a través del proceso de carga de archivos, procesamiento y visualización de resultados. Cada sección de la aplicación está cuidadosamente organizada para asegurar una experiencia de usuario fluida y comprensible.

4.5.1.4 Preprocesamiento de señales

Las señales de ECG se preprocesaron utilizando los pasos descritos en la sección 4.2.3, los cuales incluyen eliminación de ruido, normalización Z-score, detección de picos R y segmentación de señales.

4.5.1.5 Carga del modelo

El modelo de red recurrente ajustado con hiperparámetros, que mostró los mejores resultados en las métricas de evaluación, se cargó en el archivo Python. Este modelo es el núcleo del sistema de predicción, analizando las señales de ECG preprocesadas y generando predicciones sobre la presencia de arritmias. Las predicciones incluyen la determinación de si la alarma es falsa o verdadera y el tipo de arritmia, entre las que se encuentran asistolia, bradicardia, taquicardia, fibrilación/aleteo ventricular y taquicardia ventricular.

4.5.1.6 Interfaz de usuario

La interfaz web de usuario fue diseñada para ser fácil de usar y altamente funcional. Permite a los usuarios cargar los archivos de señal de ECG (.hea y .mat), procesar estos archivos y visualizar los resultados de las predicciones de manera clara. La interfaz incluye:

- Campos de carga de archivos: Los usuarios pueden cargar archivos directamente desde su dispositivo.
- Mensajes de estado: La aplicación muestra mensajes de estado en tiempo real para mantener a los usuarios informados sobre el progreso del procesamiento.
- Visualización de resultados: Los resultados de las predicciones se muestran de manera estructurada, indicando si la alarma es verdadera o falsa y el tipo de arritmia detectada.

4.5.2 Manual de Uso

4.5.2.1 Acceso a la interfaz web

Una vez iniciada la aplicación, se genera una URL pública mediante Ngrok. Copie y pegue esta URL en su navegador web para acceder a la interfaz.

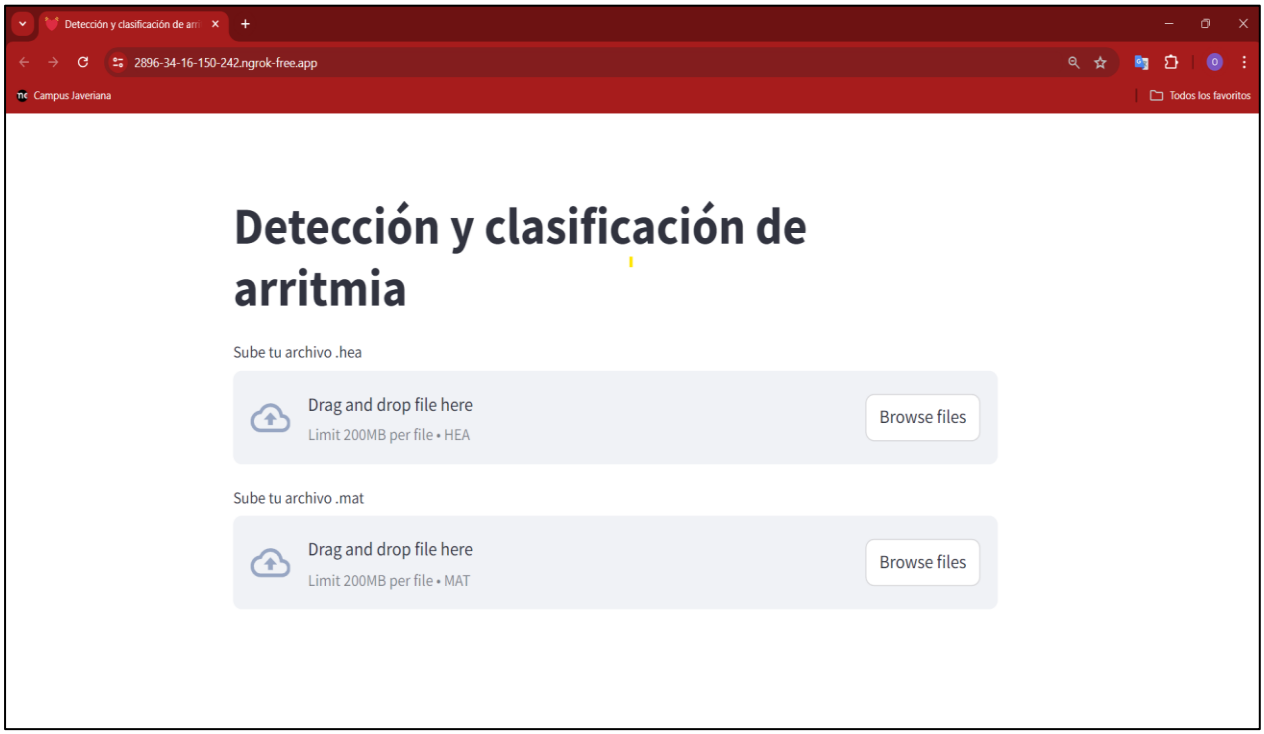

Fig. 12 Interfaz web pantalla inicial

4.5.2.2 Carga de Archivos

- Al abrir la aplicación, encontrará dos campos claramente etiquetados para cargar archivos: uno para archivos `.hea` y otro para archivos `.mat`.
- Haga clic en "Browse files" y seleccione los archivos correspondientes desde su dispositivo. Ambos archivos son necesarios para realizar la predicción.

4.5.2.3 Procesamiento y Predicción

- Después de cargar los archivos, la aplicación comenzará a leer y procesar las señales de ECG automáticamente. No es necesario realizar ninguna acción adicional.
- El estado del procesamiento se mostrará en la pantalla (por ejemplo, "Leyendo archivos...", "Preprocesando señal..."). Estos mensajes ayudan a entender en qué etapa del proceso se encuentra la aplicación.
- Una vez que el procesamiento se complete, la aplicación mostrará los resultados de las predicciones.

4.5.2.4 Visualización de Resultados

- La sección "Etiquetas reales de la señal" mostrará las etiquetas originales presentes en el archivo `.hea`, proporcionando un contexto sobre la señal analizada.
- La sección "Resultados de predicción" mostrará si la alarma es verdadera o falsa y el tipo de arritmia detectada, si corresponde. Esto incluye una interpretación fácil de entender de los resultados obtenidos del modelo de aprendizaje profundo.

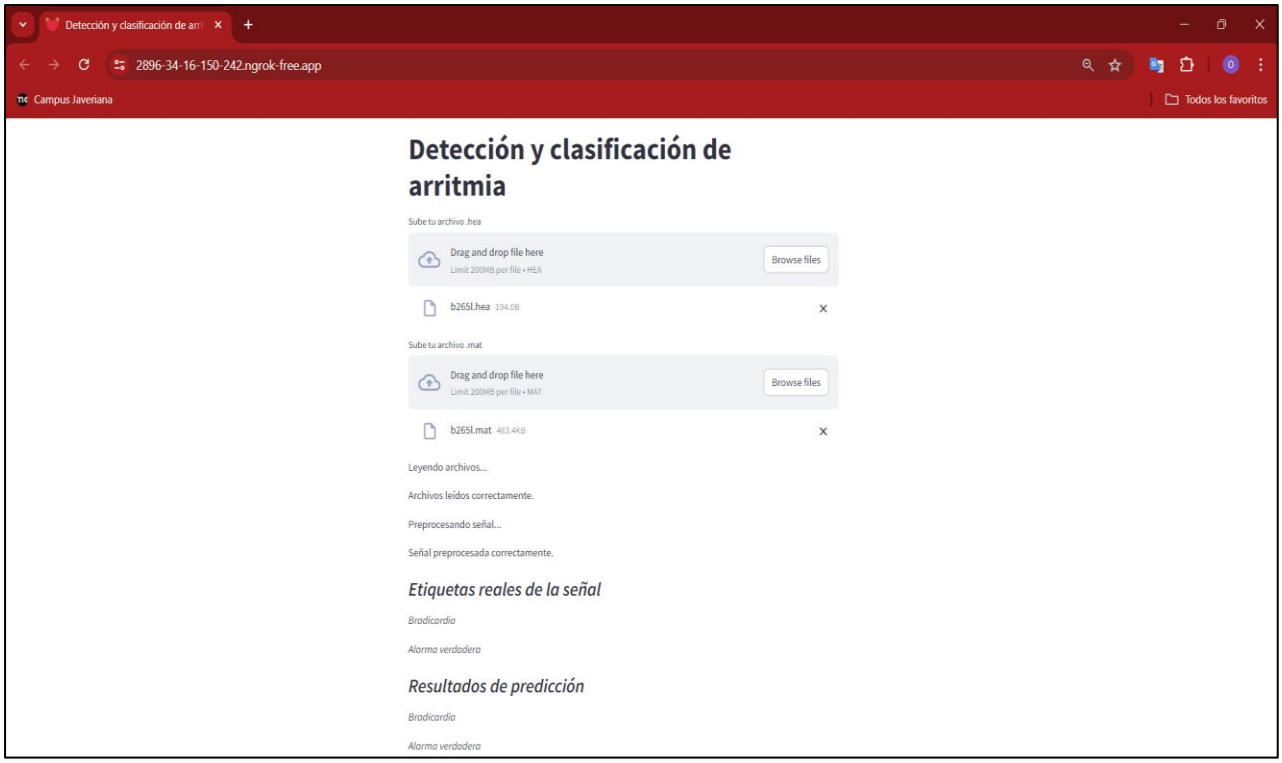

Fig. 13 Interfaz web con la visualización de resultados

4.5.2.5 Errores de usuario y Soluciones

- Si ocurre algún error durante el procesamiento, este se mostrará en la pantalla. Asegúrese de que los archivos cargados estén en el formato correcto y que contengan los datos necesarios.
- En caso de problemas con la conexión de Ngrok, verifique su token de autenticación y asegúrese de que no haya otros túneles abiertos. Si el problema persiste, puede reiniciar la aplicación para resolver conflictos de conexión.

5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1. CONCLUSIONES

El desarrollo y la evaluación de este proyecto han permitido clasificar arritmias cardiacas en señales de electrocardiograma (ECG) mediante técnicas de aprendizaje profundo. Se logró preparar un conjunto de datos de señales de ECG con arritmias cardiacas, extrayendo y preprocesando 740 señales del canal II de la base de datos *'Reducing False Arrhythmia Alarms in the ICU: The PhysioNet/Computing in Cardiology Challenge 2.015'*. De los 750 registros disponibles, se encontraron 10 que no contenían el canal II, lo cual se tuvo en cuenta para garantizar la calidad y consistencia del conjunto de datos. Durante la preparación de los datos, se llevaron a cabo múltiples pasos de preprocesamiento, incluyendo la eliminación de ruido mediante filtros de paso bajo, paso alto, mediana y Kalman, y la normalización Z-Score. Esto permitió mejorar la calidad de las señales de ECG, facilitando una mejor extracción de características y rendimiento de los modelos.

Se implementaron dos modelos de aprendizaje profundo: una red neuronal convolucional (CNN) y una red neuronal recurrente (RNN). La arquitectura del modelo CNN incluyó 14 capas en total, compuestas por capas convolucionales, de max-pooling, de aplanado, densas y de dropout, resultando en un total de 386.055 parámetros. La arquitectura del modelo RNN consistió en 14 capas, incluyendo múltiples capas LSTM, capas de dropout y densas, con un total de 99.975 parámetros. El ajuste de hiperparámetros se realizó mediante la técnica de búsqueda aleatoria utilizando Keras Tuner, optimizando parámetros como el número de filtros en las capas convolucionales, el número de unidades en las capas densas y la tasa de dropout. Los hiperparámetros óptimos encontrados permitieron mejorar significativamente las métricas de precisión y sensibilidad en ambos modelos.

La evaluación de los modelos mostró que ambos, CNN y RNN, mejoraron significativamente después del ajuste de hiperparámetros. El modelo RNN ajustado alcanzó una precisión de 97,36% para la clasificación de alarmas y 92,10% para la clasificación de arritmias. En particular, el modelo RNN ajustado mostró un desempeño destacado en la clasificación de Taquicardia, con una precisión del 97%, y en la clasificación de Taquicardia Ventricular, con una sensibilidad del 96%. El modelo CNN ajustado también mostró mejoras notables, alcanzando una precisión de 96,69%

para la clasificación de alarmas y 91,17% para la clasificación de arritmias. Destacó en la clasificación de Taquicardia con una precisión del 97% y en la clasificación de Bradicardia con una F1-Score del 90%.

Al comparar las arritmias con los resultados más elevados y más bajos, se observó que la Taquicardia y la Taquicardia Ventricular fueron las mejor clasificadas en ambos modelos, posiblemente debido a su mayor frecuencia en los datos. Por otro lado, la Aleteo/Fibrilación Ventricular presentó métricas de precisión y sensibilidad más bajas, lo que puede atribuirse a su menor frecuencia en los registros disponibles. Estos resultados sugieren que la frecuencia de las arritmias influye en el desempeño del modelo, destacando la importancia de tener un balance adecuado para mejorar la clasificación de todas las arritmias.

Se desarrolló una interfaz gráfica de usuario basada en el modelo RNN ajustado, que mostró los mejores resultados en las métricas de evaluación. La interfaz se implementó utilizando la biblioteca Streamlit, permitiendo la carga, procesamiento y visualización de las etiquetas reales de las señales de ECG de manera interactiva. Los usuarios pueden cargar archivos de señal de ECG, procesar estos archivos y visualizar los resultados de las predicciones sobre si la alarma es falsa o verdadera y el tipo de arritmia (Asistolia, Bradicardia, Taquicardia, Aleteo/Fibrilación Ventricular, Taquicardia Ventricular). La interfaz amigable y los procesos automatizados aseguran que el sistema pueda ser utilizado sin necesidad de conocimientos técnicos avanzados.

5.2. TRABAJOS FUTUROS

Para continuar y expandir los logros de este proyecto, se proponen las siguientes líneas de trabajo futuro:

- Explorar y evaluar otras arquitecturas de redes neuronales, como las redes neuronales convolucionales profundas (Deep CNN) o híbridas que combinen CNN y RNN, para mejorar aún más la precisión y sensibilidad en la clasificación de arritmias.

- Ampliar el conjunto de datos utilizado para el entrenamiento y evaluación de los modelos, incluyendo señales de ECG de diferentes bases de datos. Un conjunto de datos más amplio y diverso podría mejorar la generalización del modelo y su rendimiento en situaciones del mundo real.

- Realizar estudios de validación clínica para evaluar el desempeño del modelo en entornos reales. Involucrar a profesionales de la salud para obtener retroalimentación y ajustar el modelo según sea necesario para asegurar su aplicabilidad y efectividad en la práctica médica.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] OMS, «La OMS revela las principales causas de muerte y discapacidad en el mundo: 2000-2019,» Organización Panamericana de la Salud, 10 12 2020. [En línea]. Available: https://www.paho.org/es/noticias/9-12-2020-oms-revela-principales-causas-muerte-discapacidad-mundo-2000-2019. [Último acceso: 28 05 2023].

[2] Á. A. García Peña, D. Ospina, J. Rico, D. G. Fernández Ávila, Ó. Muñoz Velandia y F. Suárez Obando, «Prevalencia de hipertensión arterial en Colombia según información del Sistema Integral de Información de la Protección Social (SISPRO),» Revista Colombiana de Cardiología, 2022 02 2022. [En línea]. Available: https://publicaciones.revcolcard.site/articulo/11497. [Último acceso: 2023 05 28].

[3] OMS, «Enfermedades Cardiovasculares,» Organización Mundial de la Salud, [En línea]. Available: https://www.who.int/es/health-topics/cardiovascular-diseases#tab=tab_1. [Último acceso: 28 05 2023].

[4] OPS, «La carga de las enfermedades cardiovasculares en la Región de las Américas, 2000-2019,» Portal de Datos de NMH, 2021. [En línea]. Available: https://www.paho.org/es/enlace/carga-enfermedades-cardiovasculares. [Último acceso: 2023 05 28].

[5] Mayo Clinic, «Arritmia cardíaca,» Mayo Clinic, [En línea]. Available: https://www.mayoclinic.org/es-es/diseases-conditions/heart-arrhythmia/symptoms-causes/syc-20350668.

[6] J. A. Zavala Villeda, «Descripción del electrocardiograma normal y lectura del electrocardiograma,» *Revista Mexicana de Anestesiología,* vol. 40, pp. 210-213, 2017.

[7] J. L. Reis Silva y A. Tomazati Oliveira, «Classificador Automático de Arritmias Cardíacas Usando Técnicas de TinyML Aplicado a Sistemas Embarcados,» de *Educação, Ciência, Tecnologia e Sociedade [livro eletrônico]* , Belo Horizonte, FoxTablet, 2022, pp. 19 - 28.

[8] J. Zheng, H. Guo y H. Chu, «A large scale 12-lead electrocardiogram database for arrhythmia study,» PhysioNet, 24 08 2022. [En línea]. Available: https://doi.org/10.13026/wgex-er52.

[9] J. C. Bocanegra Rivera, «Sobrecarga laboral en los profesionales de la salud y su relación con la seguridad del paciente: la fatiga resultante de la sobrecarga en el trabajo produce un deterioro del rendimiento humano, el cual puede verse reflejado en errores de omisión,» Gale OneFile, 04 2012. [En línea]. Available: https://go.gale.com/ps/i.do?p=IFME&u=googlescholar&id=GALE|A307524648&v=2.1&it=r&sid=IFME&asid=1e084623. [Último acceso: 07 06 2023].

[10] D. P. Rodríguez, «Actualmente en Colombia solo hay un médico especialista por cada 1.000 habitantes,» La República, 23 05 2023. [En línea]. Available: https://www.larepublica.co/economia/actualmente-en-colombia-solo-hay-un-medico-especialista-por-cada-1-000-habitantes-3609487.

[11] F. Alonso Atienza, Estudio de los mecanismos de las arritmias cardiacas mediante modelado y procesado robusto digital de señal, Tesis de Doctorado, Leganés: Universidad Carlos III de Madrid, 2008.

[12] Texas Heart Institute, «Anatomía del corazón,» Texas Heart Institute, [En línea]. Available: https://www.texasheart.org/heart-health/heart-information-center/topics/anatomia-del-corazon/. [Último acceso: 05 06 2023].

[13] American Heart Association, «What is an Arrhythmia?,» American Heart Association, Inc, [En línea]. Available: https://www.heart.org/en/health-topics/arrhythmia/about-arrhythmia. [Último acceso: 05 06 2023].

[14] G. D. Clifford, I. Silva, B. Moody, Q. Li, D. Kella, A. Shahin, T. Kooistra, D. Perry y R. G. Mark, «The PhysioNet/Computing in Cardiology Challenge 2015: Reducing False Arrhythmia Alarms in the ICU,» *In 2015 Computing in Cardiology Conference (CinC),* vol. 101, pp. 273-276, 15 02 2015.

[15] P. S. Addison, «Wavelet transforms and the ECG: A review,» *Physiological Measurement,* vol. 26, nº 5, p. R155–R199, 2005.

[16] M. F. Cabrales Neira y D. I. Vanegas Cadavid, Manual de métodos diagnósticos en electrofisiología cardiovascular, Sociedad Colombiana de Cardiología y Cirugía Cardiovascular, 2006.

[17] A. López Farré y C. Macaya Miguel, Libro de la salud cardiovascular del Hospital Clínico San Carlos y de la Fundación BBVA, Bolbao, España: Fundación BBVA, 2009, p. 696.

[18] L. Schamroth, An Introduction to Electrocardiography, Blackwell Publishing, 2003.

[19] E. O'Brien, ABC of Hypertension, BMJ Books, 2013.

[20] M. J. Tobin, Principles and Practice of Mechanical Ventilation, McGraw-Hill Education, 2013.

[21] J. G. Webster, Design of Pulse Oximeters, Taylor & Francis, 1997.

[22] M. Z. Poh, N. C. Swenson y R. W. Picard, «A wearable sensor for unobtrusive, long-term assessment of electrodermal activity,» *IEEE Transactions on Biomedical Engineering,* vol. 57, nº 5, pp. 1243 - 1252, 2010.

[23] J. Pan y W. J. Tompkins, «A Real-Time QRS Detection Algorithm,» *IEEE Transactions on Biomedical Engineering,* Vols. %1 de %2BME-32, nº 3, pp. 230 - 236, 1985.

[24] P. Laguna, R. Jané y P. Caminal, «Adaptive filtering of ECG Baseline Wander,» *IEEE Engineering in Medicine and Biology Magazine,* vol. 18, nº 6, pp. 74-83, 1999.

[25] Z. M. Abdullah Al , K. Thapa y S. H. Yang, «Improving R Peak Detection in ECG Signal Using Dynamic Mode Selected Energy and Adaptive Window Sizing Algorithm with Decision Tree Algorithm,» *Sensors,* vol. 21, nº 19, p. 6682, 2021.

[26] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[27] T. Hastie, R. Tibshirani y J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2009.

[28] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature,* vol. 521, p. 436–444, 2015.

[29] A. M. A. R. &. H. G. Graves, «Speech recognition with deep recurrent neural networks,» *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645-6649, 2013.

[30] M. García Villanueva y L. Romero Muñoz, «Diseño de una Arquitectura de Red Neuronal Convolucional para la clasificación de objetos,» *Ciencia Nicolaita,* vol. 81, pp. 46-61, 2020.

[31] C. I. Orozco, . E. Xamena y M. E. Buemi, «Human Action Recognition in Videos using a Robust CNN LSTM Approach (Reconocimiento de Acciones Humanas en Videos usando una Red Neuronal CNN LSTM Robusta),» *Revista de Ciencia y Tecnología,* vol. 20, pp. 23-36, 2020.

[32] L. Merino Ulizarna, Artist, *Uso de una red neuronal convolucional para la detección de cáncer de tipo melanoma (Proyecto Fin de Carrera).* [Art]. Universidad Politécnica de Madrid, 2022.

[33] P. Loncomilla, «Deep learning: Redes convolucionales,» 2016. [En línea]. Available: https://ccc.inaoep.mx/~pgomez/deep/presentations/2016Loncomilla.pdf. [Último acceso: 15 Marzo 2024].

[34] H. L. R. P. A. Jasper Snoek, «Practical bayesian optimization of machine learning algorithms,» *Advances in Neural Information Processing Systems,* vol. 2, 2012.

[35] T. Pérez Sosa, Y. Cruz, I. La Fé y R. Quiza, «Biblioteca de clases para optimización multiobjetivo mediante el método de entropía cruzada,» *Revista Cubana de Ciencias Informáticas,* vol. 13, nº 1, pp. 90-104, 2019.

[36] D. M. Powers, «Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,» *Journal of Machine Learning Technologies,* pp. 37-63, 2011.

[37] R. Borja Robalino, A. Monleon Getino y J. Rodellar, «Estandarización de métricas de rendimiento para clasificadores Machine y Deep Learning,» *Revista Ibérica de Sistemas e Tecnologias de Informação,* vol. 30, nº 06, pp. 184-196, 2020.

[38] A. Zapeta Hernández, G. A. Galindo Rosales, H. J. Juan Santiago y M. Martínez Lee, «Métricas de rendimiento para evaluar el aprendizaje automático en la clasificación de imágenes petroleras utilizando redes neuronales convolucionales,» *Ciencia Latina Revista Científica Multidisciplinar,* vol. 6, nº 5, pp. 4624-4637, 2022.

[39] P. G. Mar Hernández, P. L. Ibarra Angulo, J. C. Grijalva Acuña y J. H. Abril García, «Image recognition software geometry with Python and OpenCV,» *Journal Computer Technology,* vol. 7, nº 19, pp. 1-9, 2023.

[40] Y. Mamani Arce, J. Chacaliaza Ricaldi, L. Pelaez, O. Lanchipa R y J. A. Ramos Guivar, «Solución analítica y numérica de la ecuación de Laplace utilizando Python,» *Revista de Investigación de Física* , vol. 24, nº 2, pp. 40-48, 2021.

[41] J. Llerena Izquierdo, Codifica en Python, Cuenca-Ecuador : Universidad Politécnica Salesiana, 2020.

[42] C. M. Pineda Pertuz, Aprendizaje automático y profundo en Python, Ra-Ma Editorial, 2022.

[43] M. Herrero Agustín, Artist, *Bandidos estocásticos: introducción, algoritmos y experimentos (Trabajo de Fin de Grado).* [Art]. Universidad Complutense de Madrid, 2022.

[44] F. Quirós Pérez, Artist, *Reconocimiento de imágenes con TensorFlow desde R (Trabajo Fin de Grado).* [Art]. Universidad de Sevilla, 2022.

[45] J. Zamorano Ruiz, Artist, *Comparación y análisis de métodos de clasificación con las bibliotecas scikit-learn y TensorFlow en Python.* [Art]. Universidad de Málaga, 2019.

[46] J. Bobadilla, Machine Learning y Deep Learning. Usando python, scikit y keras, Ediciones de la U, 2021.

[47] B. J. Castillo Requiz, J. D. Tarazona Silva, C. E. Tarazona Silva, C. Hurtado Enriquez y F. A. Cornelio Orbegoso, «Automatización del análisis exploratorio de datos y procesamiento geoquímico univariado empleando Python,» *Revista del Instituto de investigación de la Facultad de minas, metalurgia y ciencias geográficas,* vol. 26, nº 51, p. e24493, 2023.

[48] G. I. Ortega Rodríguez , Artist, *Visualización volumétrica de cubos de datos espectroscópicos (Tesis Pregrado).* [Art]. Universidad Técnica Federico Santa María, 2016.

[49] . D. Pinilla García, Artist, *Visualización de perfiles de luminosidad de galaxias a alto desplazamiento al rojo con Streamlit (Trsis Maestría).* [Art]. Universidad de Valladolit, 2023.

[50] E. Fúnez Fernandez, Artist, *Diseño de un sistema de seguridad en el hogar basado en IoT y creación de prototipo (Trabajo Fin de Grado).* [Art]. Universidad de Jaén, 2022.

[51] . I. N. Marrero, Artist, *Publicación: Detección de anomalías cardiacas mediante algoritmos de inteligencia artificial (Trabajo Fin de Grado).* [Art]. Universidad Politécnica de Cartagena, 2022.

[52] N. Díaz Rey, Artist, *Detección de anomalías de COVID-19 en radiografías de tórax (Trabajo Fin de Grado).* [Art]. Universidad de Alicante, 2022.

[53] C. A. Villagrán Fuentes, «Clasificación automática de latidos de un electrocardiograma utilizando aprendizaje profundo (Tesis de Maestría),» Universidad de Concepción , Concepción, 2017.

[54] F. A. Hernández Rueda, Clasificación de registros ECG de corta duración usando redes neuronales recurrentes, Bucaramanga, Colombia: Tesis de maestría, Universidad Pontificia Bolivariana, 2020.

[55] M. L. Bergamini y S. J. Liberczuk, «Detección de patologías en señales biomédicas mediante técnicas de machine learning,» *In XXII Workshop de Investigadores en Ciencias de la Computación,* 2020.

[56] J. Blanco Prieto, Detección, clasificación e interpretación de patologías de la conducción cardíaca mediante el uso de técnicas de deep learning, Tesis de maestría, Universidad de Oviedo, 2021.

[57] C. F. Patiño Zambrano, Dispositivo vestible inteligente para la generación de alertas tempranas de eventos cardiovasculares de riesgo, Envigado: Tesis de maestría, Universidad EIA, 2022.

[58] J. Pan y W. . J. Tompkins, «A real-time QRS detection algorithm,» *IEEE Transactions on Biomedical Engineering,* Vols. %1 de %2BME-32, nº 3, pp. 230-236, 1895.

[59] F. R. Pebriansyah, P. Turnip, N. S. Syafei, A. Trisanto y A. Turnip, «Design of Arrhythmia Early Detection Interface Using Laravel Framework,» *International Conference on Artificial Intelligence and Mechatronics Systems (AIMS),* pp. 28-30, 2021.

[60] IBM, «¿Qué son las redes neuronales recurrentes?,» IBM, [En línea]. Available: https://www.ibm.com/es-es/topics/recurrent-neural-networks. [Último acceso: 06 07 2023].

[62] D. Patiño, J. Medina, R. Silva, A. Guijarro y J. Rodríguez, «Predicción de arritmias e infartos agudos de miocardio usando aprendizaje automático,» *Revista de Ciencia y Tecnología,* 2023.

[63] G. D. Clifford, I. Silva, B. Moody, Q. Li, . D. Kella, A. Shahin, T. Kooistra, D. Perry y R. G. Mark, «Reducing False Arrhythmia Alarms in the ICU: The PhysioNet/Computing in Cardiology Challenge 2015,» *IEEE,* nº In 2015 Computing in Cardiology Conference (CinC), pp. 273-276, 2015.

[64] C. Hernández y J. Rodríguez, «PREPROCESAMIENTO DE DATOS ESTRUCTURADOS,» *Rev. Vínculos,* vol. 4, p. 27–48, 2013.

[65] R. Tandra y A. Sahai, «SNR Walls for Signal Detection,» *IEEE Journal of Selected Topics in Signal Processing,* vol. 2, pp. 4-17, 2008.

[66] Z. Zhao y Y. Zhang, «SQI Quality Evaluation Mechanism of Single-Lead ECG Signal Based on Simple Heuristic Fusion and Fuzzy Comprehensive Evaluation,» *Sec. Computational Physiology and Medicine,* vol. 9, 2018.

[67] O. Köksoy, «Multiresponse robust design: Mean square error (MSE) criterion,» *Applied Mathematics and Computation [Online],* vol. 175, pp. 1716-1729, 2006.

[68] Í. Corera Orzanco, Procesado de señales de Scanning-EMG (Tesis de Maestría), Escuela Técnica Superior De Ingenieros Industriales y de Telecomunicación, 2015.

[69] J. Huircán, Filtros activos, conceptos básicos y diseño, Araucanıa, Chile: Departamento de Ingenierıa Eléctrica, Universidad de La Frontera, 2012.

[70] J. Fernández Bes, V. Elvira y S. Van Vaerenbergh, « probabilistic least-mean-squares filter,» *In 2015 IEEE International Conference on Acoustics,* pp. 2199-2203, 2015.

[71] M. Vidaurrázaga y L. Rodríguez, «Filtrado de la mediana en el mejoramiento de imagenes medicas basado en la transformada wavelet,» *Ingeniería Electrónica, Automática y Comunicaciones,* vol. 22, pp. 41-47, 2001.

[72] P. Diniz, The least-mean-square (LMS) algorithm. Adaptive Filtering: Algorithms and Practical Implementation, 2020.

7. ANEXOS

Anexo A: Código para el entendimiento de los datos

Descripción: Este anexo contiene el código utilizado para el entendimiento inicial de los datos del proyecto. El código incluye la carga de archivos desde Google Drive, la lectura y procesamiento de archivos .hea y .mat, y la visualización de una muestra aleatoria de los registros. El código se encuentra en el siguiente repositorio: https://bit.ly/45Lwi8A

Anexo B: Código para la extracción y guardado del Canal II y etiquetas

Descripción: Este anexo contiene el código utilizado para la extracción y guardado del canal II y sus etiquetas desde los archivos .hea y .mat. El proceso incluye la lectura de archivos, la identificación del canal II, la extracción de la señal y el almacenamiento de los datos y etiquetas en formato .npy. El código se encuentra en el siguiente repositorio: https://bit.ly/3VUus1Q

Anexo C: Código para la implementación de la Red Convolucional

Descripción: Este anexo contiene el código utilizado para la implementación de la red neuronal convolucional (CNN). El código abarca desde la carga y normalización de las señales, hasta la construcción y entrenamiento del modelo CNN. También incluye técnicas de filtrado de la señal, cálculo de picos R, y evaluación del rendimiento del modelo. El código se encuentra en el siguiente repositorio: https://bit.ly/4eyUBdN

Anexo D: Código para la implementación de la Red Recurrente

Descripción: Este anexo contiene el código utilizado para la implementación de la red neuronal recurrente (RNN). El código abarca desde la carga y normalización de las señales, hasta la construcción y entrenamiento del modelo RNN. También incluye técnicas de filtrado de la señal, cálculo de picos R, y evaluación del rendimiento del modelo. El código se encuentra en el siguiente repositorio: https://bit.ly/4cziNLj

Anexo E: Código para la creación de la Interfaz Web

Descripción: Este anexo contiene el código utilizado para la creación de la interfaz web del proyecto. El código incluye la instalación de dependencias, configuración de Streamlit y Ngrok, y la implementación de la interfaz que permite la carga y visualización de datos, así como la interacción con los modelos de detección y clasificación de arritmias. El código se encuentra en el siguiente repositorio: https://bit.ly/3XBBQQK