COMPRESIÓN DE MODELOS BASADOS EN APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE LA FIBRILACIÓN AURICULAR

JEYSON ARLEY CASTILLO BOHÓRQUEZ

Tesis de investigación presentada como requisito parcial para optar al título de:

Magíster en Ingeniería Electrónica

Director:

Carlos Fajardo Ariza

Ph.D. en Ingeniería Electrónica.

UNIVERSIDAD INDUSTRIAL DE SANTANDER

ESCUELA DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES

BUCARAMANGA, COLOMBIA

2026

Dedicatoria

Este trabajo viene dedicado para todas aquellas personas que apoyaron el desarrollo y ejecución de este trabajo de grado.

En especial reconocer la permanente presencia de Dios en el camino de nuestra vida.

A mis padres

A mi Director

¡Levanta la mirada! Si te sientes desalentado porque crees que no eres suficiente, enciende la llama de tu corazón. Sécate los ojos y mira hacia adelante. Puede que quieras frenar, pero el paso del tiempo no espera a nadie, no te hará compañía ni compartirá tus penas. mantén tu corazón ardiendo.

- Rengoku Kyojuro

Agradecimientos

A mis padres que ni por un segundo dudaron de mis capacidades y lo que puedo llegar a lograr. A mi padre por enseñarme a aprender, y a mi madre por enseñarme a enseñar. A mi director, que más que un guía académico, fue un amigo en medio de un mar tormentoso, un consejero en el camino oscuro, y una luz para no olvidar a Dios. Al profesor Gabriel, porque sin esa llamada, probablemente no hubiese retomado nunca más el maravilloso sendero del conocimiento. Al grupo CPS por ser más que un grupo, una comunidad abierta de apoyo y guía.

Contenido

Lista de Tablas

Lista de Figuras

Resumen

Título: Compresión de modelos basados en aprendizaje profundo para la detección de la fibrilación auricular*

Autor: Jeyson Arley Castillo Bohórquez**

Palabras clave: Fibrilación auricular, computación en el borde, detección automática, arritmia cardíaca, red neuronal convolucional, aprendizaje profundo, ECG, dispositivo portable.

Descripción:

En la actualidad, el acceso a servicios de salud especializados enfrenta retos asociados al crecimiento poblacional, la disponibilidad de personal médico y la necesidad de diagnósticos oportunos. Entre las enfermedades de mayor impacto se encuentran las cardiovasculares, especialmente las arritmias cardíacas como la fibrilación auricular, cuyo monitoreo continuo resulta fundamental para mejorar la calidad de vida de los pacientes y reducir riesgos clínicos. En este contexto, los sistemas portátiles de monitoreo y la inteligencia artificial ofrecen una alternativa prometedora para apoyar la detección automática de este tipo de alteraciones a partir de señales electrocardiográficas.

Este trabajo implementa un conjunto de modelos de redes neuronales profundas comprimidos para la detección de fibrilación auricular. Se desarrollan arquitecturas convolucionales compactas orientadas a mantener un desempeño competitivo con un número reducido de parámetros. Además, se aplican estrategias de compresión como compactación, poda, cuantización y destilación de conocimiento, con el propósito de disminuir los requerimientos de memoria y la complejidad computacional de los modelos. De manera complementaria, se propone el uso de aprendizaje contrastivo para mejorar el rendimiento en escenarios con pocos datos etiquetados, modelos pequeños y presencia de ruido en las señales.

Los resultados se analizan en términos de precisión, tamaño de memoria y operaciones en punto flotante, considerando el equilibrio entre desempeño predictivo y eficiencia computacional. Asimismo, se comparan con estudios relacionados que emplean redes neuronales profundas para la detección de fibrilación auricular. Los hallazgos sugieren que es posible desarrollar modelos de baja complejidad capaces de detectar esta arritmia con buen desempeño, lo cual favorece su implementación en dispositivos portátiles y sistemas de monitoreo continuo sin comprometer significativamente la precisión de la predicción.

* Tesis de maestría

** Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería Eléctrica, Electrónica y de Telecomunicaciones. Director: Carlos Fajardo Ariza, Ph.D.

Abstract

Title: Compression of deep learning-based models for atrial fibrillation detection*

Author: Jeyson Arley Castillo Bohórquez**

Key words: Atrial fibrillation, edge computing, automatic detection, cardiac arrhythmia, convolutional neural network, deep learning, ECG, wearable device.

Description:

Currently, access to specialized healthcare services faces challenges associated with population growth, the availability of medical personnel, and the need for timely diagnosis. Among the diseases with the greatest impact are cardiovascular conditions, especially cardiac arrhythmias such as atrial fibrillation, whose continuous monitoring is essential to improve patients' quality of life and reduce clinical risks. In this context, portable monitoring systems and artificial intelligence provide a promising alternative to support the automatic detection of this type of alteration from electrocardiographic signals.

This work implements a set of compressed deep neural network models for atrial fibrillation detection. Compact convolutional architectures are developed to maintain competitive performance with a reduced number of parameters. In addition, compression strategies such as model compactness, pruning, quantization, and knowledge distillation are applied to reduce memory requirements and computational complexity. As a complementary approach, contrastive learning is proposed to improve model performance in scenarios involving limited labeled data, small models, and noisy signals.

The results are analyzed in terms of accuracy, memory size, and floating-point operations, considering the balance between predictive performance and computational efficiency. They are also compared with related studies that use deep neural networks for atrial fibrillation detection. The findings suggest that it is possible to develop low-complexity models capable of detecting this arrhythmia with strong performance, which supports their implementation in wearable devices and continuous monitoring systems without significantly compromising prediction accuracy.

* Master's thesis

** Faculty of Physical-Mechanical Engineering. School of Electrical, Electronic and Telecommunications Engineering. Director: Carlos Fajardo Ariza, Ph.D.

Introducción

La fibrilación auricular (FA) es la arritmia cardiaca más frecuente en la práctica clínica y se asocia a una menor calidad de vida y aumenta el riesgo de ictus e infarto de miocardio. La prevalencia global de la FA es de alrededor del 2 % en la comunidad general, cerca del 5 % en pacientes mayores de 60 años y del 10 % en personas mayores de 80 años (Forero-Gómez et al., 2017) ,(Romero & Chávez, 2014). En Colombia, algunos estudios muestran que la incidencia y la mortalidad de la FA han aumentado especialmente en personas mayores de 70 años (Romero & Chávez, 2014), (Castillo et al., 2020). La FA es generalmente asintomática o presenta síntomas inespecíficos (Bhatt & Fischer, 2015). Sin embargo, varios informes médicos llaman la atención sobre la importancia del diagnóstico previo para iniciar un tratamiento temprano que mejore la calidad de vida de los pacientes (Čihák et al., 2016), (Galvez-Olortegui et al., 2016). Para poder realizar un diagnóstico correcto y determinar su tratamiento se requiere el monitoreo constante según la variación en la enfermedad que se presente en el paciente. De acuerdo a la práctica clínica la FA categorizada en cuatro clases que son i) paroxística: episodios menores a 48 horas que terminan sin intervención clínica, ii) persistente: episodios que superan los 7 días y se requiere de cardioversión para terminarla. iii) persistente de largo termino: duración superior a un año y iv) permanente: alteración total del ritmo sinusal y no es posible restaurarlo con cardioversión. Catalogarla requiere de periodos de monitorio extensos que ayuden a clasificar correctamente la enfermerdad.

Una de las maneras mas sencillas de monitorear esta enfermedad es mediante electrocardiografía. El uso de electrodos colocados cerca del pecho, la actividad eléctrica del corazón puede detectarse y registrarse como señales electrocardiográficas. La morfología de las ondas del ECG

contiene información característica sobre las condiciones del corazón. Basándose en los patrones de las ondas de estos cambios eléctricos, los cardiólogos realizan diagnósticos de arritmias y otras enfermedades del corazón (Acharya et al., 2017). Por otra parte una vez realizado el monitorero se requiere que los médicos evalúen estas señales por largos periodos de tiempo lo que hace a este tipo de diagnóstico una actividad bastante ardua y meticulosa, la cual consume mucho tiempo y recurso humano.

Las Redes Neuronales se convierten en una herramienta útil para el diagnóstico médico de este tipo de señales, por su amplia capacidad en problemas de clasificación y extracción de características. Una red neuronal entrenada para etiquetar la señal podria disminuir considerablemente los tiempos requeridos para el diagnóstico de esta enfermedad. Estudios recientes muestran cómo las Redes Neuronales Convolucionales (CNN) y otros tipos de Redes Neuronales Profundas (DNN) se han utilizado para detectar arritmias cardíacas con una precisión muy alta (Datta et al., 2017), (Hong et al., 2017), (Teijeiro et al., 2017), (Zabihi et al., 2017), (Mahajan et al., 2017), (Hannun et al., 2019a). La mayoría de los investigadores se han centrado principalmente en mejorar la precisión. Para cualquier problema de clasificación, múltiples arquitecturas de CNN logran niveles similares de precisión, por ejemplo, en el Physionet Challenge de 2017 varias arquitecturas lograron una puntuación F1 superior a 0,80 (Clifford et al., 2017). La mayoría de los desarrollos de Deep Learning para la clasificación de arritmias implican modelos complejos sobredimensionados. En consecuencia, el despliegue en dispositivos de memoria limitada es difícil debido a los grandes requisitos de ancho de banda y memoria (Glorot & Bengio, 2010), (Erhan et al., 2009).

En el entrenamiento distribuido de datos en paralelo, la sobrecarga de comunicación es directamente proporcional al número de parámetros del modelo. Como resultado, el proceso de entrenamiento requiere una gran memoria caché o mejores requisitos de clústeres paralelos. Para el cálculo en paralelo, los dispositivos de hardware, como los FPGA, son mejores que los microcontroladores, pero los FPGA suelen tener menos de 10 MB de memoria en el chip, lo que es pequeño en comparación con los tamaños típicos de los modelos. Por lo tanto, es posible entrenar modelos en un ordenador potente y desplegar sólo el proceso de inferencia en un dispositivo portátil con un

modelo más pequeño. Para ello, es necesario desarrollar modelos menos complejos con un menor número de parámetros, pero que alcancen niveles de precisión similares.

Estudios recientes han identificado un grupo de estrategias de diseño arquitectónico para reducir el tamaño y la memoria. Iandola (Iandola et al., 2016) muestra con fuertes evidencias empíricas cómo es posible comprimir un modelo hasta 50x en cuanto a los parámetros, logrando una precisión cercana al modelo original. En ese trabajo, muestran la importancia de la reducción de parámetros en relación con el despliegue de modelos en FPGAs y otro hardware con memoria limitada y explican cómo esta técnica es factible a través de tres estrategias. Por otro lado, la última estrategia trata de maximizar la precisión con un presupuesto limitado de parámetros. En otra investigación de Han et al. (Han, Pool et al., 2015), (Han, Mao & Dally, 2015) se desarrolló el Network Pruning, donde se combina la poda, la cuantificación y la codificación Huffman. Una de estas estrategias se centra en la reducción del número de operaciones, MAPs y operadores. Por otro lado, las estrategias de reducción de memoria se concentran en la representación numérica y la distribución de los pesos. (Iandola et al., 2016) (Han, Pool et al., 2015), (Han, Mao & Dally, 2015).

A través de esta investigación se han obtenido un modelo de Deep Learning para detectar Fibrilación Auricular con una reducción de parámetros notable, Por medio de la aplicación combinada de técnicas como knowledge distillation y pruning. Como resultado se cuenta con un modelo reducido de 20,289 parámetros, y que no compromete significativamente la precisión en comparación con el modelo base (arriba de 0.9 en el F1 score).

1. Estrategias de compresión

Las redes neuronales han alcanzado un gran éxito en los últimos años y actualmente se usan en muchas tareas de clasificación de imágenes, detección de objetos y procesamiento del lenguaje natural. Su enorme desarrollo viene acompañada por un sin número de retos y desafíos para su implementación en sistemas convencionales y portátiles.Hoy en día existen modelos elaborados que llegan a contener miles de millones de parámetros y que han logrado superar la capacidad de clasificación humana en tareas de alta complejidad. Sin embargo, estas redes por lo general están sobre parametrizadas e implican un alto consumo computacional y de memoria. Todo esto impide su implementación en sistemas portátiles y de bajo consumo. (Iandola et al., 2016)

En la actualidad existen estrategias que permiten reducir las redes tanto en complejidad computacional como en memoria requerida. Una de estas estrategias es la compactación de modelos que mediante el cambio de la estructura básica de la red, elimina las redundancias, logrando pérdidas mínimas en la exactitud del modelo. Otras estrategias es el *Prunning* o poda que se concentran principalmente en la eliminación de los pesos de bajo valor que producen operaciones innecesarias. La cuantización es una estrategia de la memoria y la representación de los números que componen los pesos de la red. Por otra parte existen técnicas que buscan mejorar el proceso de aprendizaje. Una de ellas desarrollada recientemente es la destilación de conocimiento (”Knowledge Distillation”) que busca que modelos más pequeños aprendan apoyados por modelos más grandes, haciendo una transferencia de conocimiento, también llamada ”Dark Knowledge”

1.1. Compactación del modelo

Uno de los propósitos principales de la compactación de modelos de redes Neuronales Profundas, es reducir el numero de operaciones que deben llevar a cabo y simplificar la cantidad de procesos sin comprometer en gran manera la arquitectura primaria de la red. Entre las redes que mejor permiten este tipo de intervención se encuentran las redes recurrentes, las redes residuales y las redes convolucionales. Se pueden aplicar tres procedimientos principales: Reducir el tamaño de los filtros, reducir la cantidad de filtros y reducir la cantidad de bloques residuales o recurrentes.

- Reducción del tamaño del kernel en las convoluciones de las redes. Las convoluciones con filtros amplios (más de 5x5) tienen a incrementar de forma notable la complejidad computacional y el tiempo de ejecución de una CNN. Como estrategia para la reducción del tamaño de los filtros se encuentran dos estrategias en el estado del arte:

  - Reducir las convoluciones grandes en varias convoluciones pequeñas consecutivas. (Iandola et al., 2016)
  - Reducir el tamaño del filtro (Szegedy et al., 2016)

  En (Iandola et al., 2016) se utilizan 3 estrategias para la reducción de parámetros y computo: La primera estrategia consiste en tomar todos los filtro de 3x3 y reducirlos a 1x1. Con esto logran reducir en 9X el numero de parámetros en las capas convolucionales. La segunda estrategia consiste en la implementación de módulos *FIRE* que permiten reducir el tamaño de los canales de entrada. Un módulo Fire está compuesto por: una capa de convolución de compresión (que sólo tiene filtros 1x1), que alimenta una capa de expansión que tiene una mezcla de filtros de convolución 1x1 y 3x3; demuestran que reducir los filtros de tamaño a 3x3 puede llegar a reducir el numero de parámetros hasta 9X. En (Szegedy et al., 2016) los autores muestran que usar filtro de 5x5 puede tomar hasta 2.78 veces mas tiempo de ejecución que si se usan filtro de 3x3. Adicionalmente indican que la reducción de parámetros en general en la red de hasta un 28 %.

- Reducción de la cantidad de filtros En SquezeeNet (Iandola et al., 2016) los autores recomiendan disminuir la cantidad de filtros en los canales de entrada esto con el fin de que los parámetros que fluyen a través de la red se mantengan en niveles bajos. En la red propuesta por los autores, mantienen los filtros de 3x3 con un similar numero de canales a los de las señales de entrada.

- Reducción de la cantidad de bloques residuales

  En ShaResNet (Boulch, 2017) los autores muestran una estrategia para la reducción virtual de los pesos sin cambiar el numero de bloques residuales. Compartir los pesos en las capas residuales hace que las redes sean casi tan eficiente como sus contrapartes secuenciales involucrando menos parámetros, y por otra parte que son más eficientes que una red residual típica con el mismo número de parámetros. Para ejemplo, una red residual de 152 capas de profundidad puede ser reducido a 106 capas convolucionales, que representa un nivel de compresión de 39 %, mientras que pierde menos del 0,2 % de precisión para la base de datos Imagenet.

En el desarrollo de este proyecto se utilizan las 3 técnicas proporcionadas, enfocándonos en la reducción del tamaño de los filtros (Szegedy et al., 2016), la reducción de la cantidad de filtros propuesto por Iandola en SquezeeNet (Iandola et al., 2016), haciendo énfasis en la técnica de las capas residuales, reduciendo el numero de repeticiones como muestra ShaResNet (Boulch, 2017).

1.2. Poda

El objetivo principal del *pruning* o poda es remover los parámetros que resultan innecesarios para el entrenamiento e inferencia de las redes neuronales, sin comprometer notablemente la precisión del modelo y reduciendo el tamaño de la misma (Blalock et al., 2020). El pruning es una técnica que ha sido usada desde finales de la década de 1980 (Mozer & Smolensky, 1988) y ha tenido un gran interés en la última Década. Actualmente en el estado del arte existen numerosos métodos para la poda de parámetros. (Xu et al., 2020) (Blalock et al., 2020). En este algoritmo, la

red se entrena primero hasta la convergencia. Después, cada parámetro o elemento estructural de la red de la red se le asigna una puntuación, y la red se poda en función de estas puntuaciones.Estas puntuaciones pueden estar basadas en los valores de los pesos, su probabilidad o su incidencia en el error. También es posible que vayan ligado al elemento estructural seleccionado para la poda.

Entrada: N número de iteraciones del pruning; X base de datos para entrenamiento

Salida: M, W

1: $W \leftarrow Inicializar()$

2: $W \leftarrow EntrenarHastaConvergencia(f(X;W))$

3: $M \leftarrow 1^{|W|}$

4: **para** i in 1 to N **hacer**

5: $\quad M \leftarrow prune(M, score(W))$

6: **fin para**

7: **devolver** W, M

Algoritmo 1: Algoritmo primario para realizar poda en modelos de Deep Learning

El algoritmo 1 representa el procedimiento básico para realizar la poda de parámetros. Nótese que en la línea 5 se menciona una función llamada *Score*; esta función representa la puntuación para aplicar la penalización a los parámetros y es de suma importancia debido a que esta regla puede ser probabilística, determinística, o estructural. Entre los tipos comunes de algoritmo encontramos la poda estructurada y la poda no estructurada.

- Poda no estructurada determinística: La poda no estructurada determinística consiste en la identificación de los parámetros que no poseen relevancia para el cálculo del error y la inferencia. Esta basado estrictamente en parámetros determinísticos estableciendo un umbral para el valor de los pesos de tal forma que se puedan forzar al valor de 0. Con esta técnica se crea una máscara de valores nulos, lo que ayuda al software a ignorar esas operaciones y conexiones pues se consideran que no tienen un aporte significativo a la propagación.

- Poda no estructurada Probabilística: Para la poda no estructurada probabilística se realiza

de manera muy similar la poda determinista pero el puntaje de decisión está dado por la probabilidad de los parámetros más cercanos a cero, de esta manera se evita la concentración de parámetros nulos que ocupan memoria.

- Poda estructurada: La poda generalmente da como resultado conexiones de red irregulares que no solo exigen esfuerzos de representación adicionales debido a las máscaras, sino que tampoco encajan bien en el cálculo paralelo. Para la poda estructurada se tiene en cuenta la arquitectura de la red. Este tipo de poda por lo general no se realiza sobre parámetros individuales, sino sobre canales de entrada, mapas característicos, filtros completos o sobre el desplazamiento de los filtros (Anwar et al., 2017; Y. He et al., 2017)

1.3. Cuantización

Mientras que algunas estrategias comprimen los modelos reduciendo el número de pesos, la cuantización consiste en transformar la representación numérica por un intervalo reducido de valores. La cuantización de un conjunto de valores distribuidos consiste en aproximar y codificar estos valores, para reducir el intervalo de salida de la representación, idealmente sin perder demasiada información en el proceso. El objetivo principal es reducir la precisión de los componentes de la red sin afectar a la exactitud. Otro objetivo es comprimir el modelo en términos de memoria volátil para mejorar el despliegue en dispositivos de memoria limitada por hardware. Se consigue reducir tanto el número de operadores lógicos necesarios como la potencia por unidad de superficie. Actualmente, existen dos formas de cuantizar un modelo de aprendizaje profundo, la primera es con entrenamiento en precisión máxima e inferencia de pocos bits, aplicando técnicas de cuantización lineal a los pesos y funciones de activación de la red neuronal (Jacob et al., 2018; Krishnamoorthi, 2018). El problema de esta técnica es que la retro-propagación no está bien definida para esta cuantización y las pérdidas de exactitud para valores amplios de reducción de bits son considerables. Otra técnica es la llamada cuantización consciente, que es una técnica que aplica una precisión muy baja a la mayoría de los datos, mientras que el resto de los datos se representa con alta precisión por separado. De este modo, se reduce el error de propagación y se consiguen niveles de precisión

muy cercanos al modelo original (Krishnamoorthi, 2018).

1.4. Destilación de conocimiento (*Knowledge Distillation*)

La primera referencia a la transferencia de conocimiento fue propuesta por los autores Caruana *et al* en (Buciluǎ et al., 2006). Ellos entrenaron un modelo comprimido de clasificadores rígidos con datos pseudo-etiquetados y lograron reproducir el resultado obtenido por una red mucho más grande. A pesar de ese logro, sus resultados estaban limitados a modelos de poca profundidad. La idea fue adoptada nuevamente bajo el nombre de *Knowledge Distillation* por los mismos autores (Ba & Caruana, 2014) donde exponen la compresión de modelos más amplios y profundos en mucho mas pequeños, donde los modelos reducidos imitaron el comportamiento de los modelos más complejos. El planteamiento principal está dado por la idea de transferir el conocimiento de un modelo grande denominado profesor a un modelo más pequeño, usando la función de probabilidad softmax. Posteriormente Hinton *et al* (Hinton et al., 2015) logra generalizar y mejorar el procedimiento. En primera instancia las redes neuronales producen probabilidades de clase utilizando una capa de salida ”softmax”que convierte la función logística, z_i, calculado para cada clase en una probabilidad, q_i comparando z_i con los otros logits.

$$q_i = \frac{exp(\frac{z_i}{T})}{\sum_{j=1}^{c} exp(\frac{z_i}{T})} \tag{1-1}$$

En la ecuación 1-1 se muestra el parámetro de la temperatura T que permite suavizar la distribución de probabilidad obtenida a la salida. Cuando T es 1 la salida corresponde a la distribución de probabilidad típica de softmax, y a medida que se hace más grande la distribución se hace más suave.

El *Knowledge distillation* consiste en la compresión del modelo enseñando a una red más pequeña a través de la comparación constante con los resultados obtenidos por el modelo más grande. Las « etiquetas blandas » o *soft labels* se refieren a los mapas de características de salida de la red más grande, son las probabilidades generadas por el profesor. La red más pequeña se entrena

entonces para aprender el comportamiento de la red más grande tratando de replicar sus resultados en cada nivel (no sólo la pérdida final). Para esto la función de pérdida que se produce en la red estudiante consta de dos partes: la primera las pérdidas producidas por las *hard labels* (etiquetas originales del *dataset*) a través de la función de crosentropía L_{ce}; la segunda por las *soft labels*, mediante la divergencia de Kullback-Leibler L_{kl} tal y como lo muestra a continuación la ecuación 1-2:

$$L_{kl} = -T^2 \sum_{i=1}^{c} p_i^t(z_i;T) \cdot log(p_i^s(z_i;T))) \tag{1-2}$$

Note que p_i^t y p_i^s son las etiquetas generadas por el profesor y el estudiante respectivamente. Finalmente, la función de pérdida completa L es una combinación lineal entre L_{kl} y L_{ce}, como sigue en la ecuación 1-3:

$$L = \alpha L_{ce} + (1-\alpha) L_{kl} \tag{1-3}$$

A partir de estos estudios y los resultados propuestos por Hinton, es interés por esta técnica se ha incrementado y nuevas investigaciones han colocado sobre la mesa nuevas formas de uso. Los más recientes métodos se han extendido al aprendizaje mutuo (Zhang et al., 2018), a la enseñanza asistida (Mirzadeh et al., 2020), al aprendizaje permanente (Zhai et al., 2019) y al auto-aprendizaje (Yuan et al., 2019). La mayoría de las extensiones de destilación de conocimientos se concentran en la compresión de redes neuronales profundas.

1.5. Datos no etiquetados: un escenario desafiante

Actualmente, en el entorno médico abundan los datos no etiquetados que se generan día tras día en los consultorios, y es muy difícil aprovecharlos por el tiempo y atención que se requiere por parte de los profesionales para su correcto etiquetado. Si bien los datos no son parte de un modelo,

si son esenciales para poder entrenarlos y de la cantidad de estos últimos dependen los resultados.

El aprendizaje automático ha tenido un gran éxito en las aplicaciones de uso intensivo de datos, pero a menudo se ve obstaculizado cuando el conjunto de datos es pequeño. Los métodos no supervisados y auto-supervisados se perfilan como una posible solución a este obstáculo. Los métodos no supervisados usan técnicas que aprovechan las características de las señales, para establecer similitudes y extraer la información oculta basada en características y no en etiquetas. Mediante este proceso se pueden crear tareas pretexto que puedan explotar los datos.

1.5.1. Aprendizaje por análisis contrastivo

El aprendizaje contrastivo es una técnica de aprendizaje automático auto-supervisado utilizada para aprender las características generales de un conjunto de datos, enseñando al modelo qué puntos de datos son similares o diferentes. La razón por la que se perfila como una técnica prometedora para mejorar la clasificación es que podemos entrenar el modelo para que aprenda sin requerir anotaciones ni etiquetas, de ahí el término, aprendizaje auto-supervisado.

En muchos de los escenarios del mundo real, no se tienen etiquetas para cada dato de interés, por ejemplo, las imágenes médicas. Para crear etiquetas, los profesionales tienen que pasar incontables horas examinando las imágenes para clasificarlas manualmente, segmentarlas, etc. Con el aprendizaje contrastivo, se puede mejorar significativamente el rendimiento del modelo incluso cuando sólo una fracción del conjunto de datos está etiquetada. El aprendizaje contrastivo puede caracterizarse por tres componentes principales:

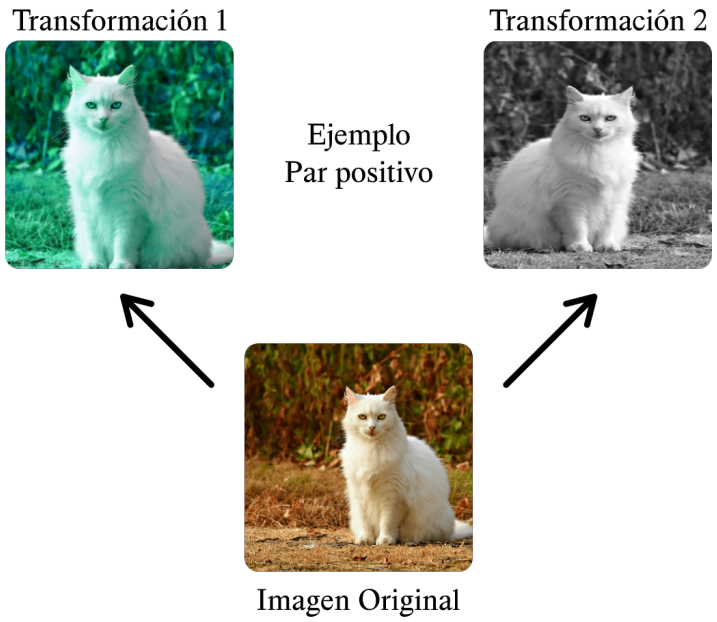
- Un conjunto de ejemplos positivos y negativos
- Un conjunto de operadores de transformación
- una variante de la pérdida de estimación contrastiva. Las técnicas de contrastive learning usualmente van ligadas a una familia de funciones de perdidas basadas en distancias vectoriales.

1.5.2. Metodología del análisis contrastivo

La primera instancia del análisis contrastivo requiere de un módulo de aumento estocástico de datos que transforma cualquier ejemplo de datos de forma aleatoria, dando lugar a dos vistas correlacionadas del mismo ejemplo. La figura a continuación muestra como se aplican transformaciones tipicas para la creación de un par positivo que alimente la red, partiendo de una imagen común (Chen et al., 2020) (Krizhevsky et al., 2012).

Figura 1

Ejemplo de como se genera un par positivo de ejemplos, usando transformaciones como: cambio de color, recorte y zoom en forma aleatoria



Nota. Adaptado de Chen et al. (Chen et al., 2020) y Krizhevsky et al. (Krizhevsky et al., 2012).

En cuanto a imágenes Chen *et al* (Chen et al., 2020) propone transformaciones como recorte, redimensión, volteo horizontal y vertical, distorsión positiva y negativa de color, rotación, ruido gaussiano desenfoque gaussiano y sombreado.Estas transformaciones en la investigación realizada por los autores demostraron mejorar el rendimiento en términos de exactitud en la base de Datos Imagenet. Por otra parte, en cuanto a datos 1D, Kiyasseh *et al* (Kiyasseh et al., 2021) muestran

un grupo de transformaciones similares sobre señales ECG, que permiten adaptar la estrategia del análisis contrastivo a problemas unidimensionales. Entre estas transformaciones propuestas están recorte, redimensión, inversión temporal y en magnitud, ruido gaussiano, máscara de tiempo y máscara en frecuencia.

2. Base de Datos y Modelo Base

2.1. Introducción

Icentia11K, es la base de datos pública más grande de señales ECG sin procesar. Contiene 11 mil pacientes y alrededor de 2 mil millones de latidos etiquetados(Tan et al., 2019). Los datos sin procesar se obtuvieron utilizando un dispositivo de monitorización cardíaca de una sola derivación, denominado CartioSTAT^{TM} (Paquet et al., 2022). Cada paciente llevó el dispositivo durante entre 3 y 14 días. Estos registros se segmentaron en segmentos de aproximadamente 70 minutos. El propósito de los creadores es proveer de una base de datos que estimule el estudio en torno a los modelos de ECG semisupervisados, así como descubrir subtipos desconocidos de arritmias y eventos de señales anómalas de ECG. Para esto proponen un porcentaje de señales etiquetadas y no etiquetadas donde el 80 % de las señales que ellos plantean como entrenamiento son No etiquetadas.

2.2. Descripción

Estas señales se tomaron de pacientes principalmente de Ontario, Canadá, con una edad media de 62,2 ± 17,4 años. La tabla 1 resume las principales características de esta base de datos.

Para reducir el tamaño del conjunto de datos, se seleccionó aleatoriamente un porcentaje de estos segmentos, que finalmente se segmentaron en tramas de 2049 muestras. Estos marcos corresponden a las señales del conjunto de datos, que tienen tres etiquetas de ritmo correspondientes a los tipos de señales: NSR (ritmo sinusal normal), AFib (fibrilación auricular) y AFlutter (aleteo auricular). En este trabajo, cada dato se etiquetó como NSR (Ritmo Sinusal Normal) (1), Arritmia Afib y Aflutter (2), o Ruido (3). Contiene 57 %, 4 % y 39 %, respectivamente. Además, filtramos

Tabla 1

Main features Icentia 11K ECG database

Estadísticas	# Cantidad
Pacientes	11,000
Latidos Etiquetados	2,774'054,978
Frecuencia de muestreo	250 Hz
Muestras por segmento	2049
Numero total de segmentos	2'555,592

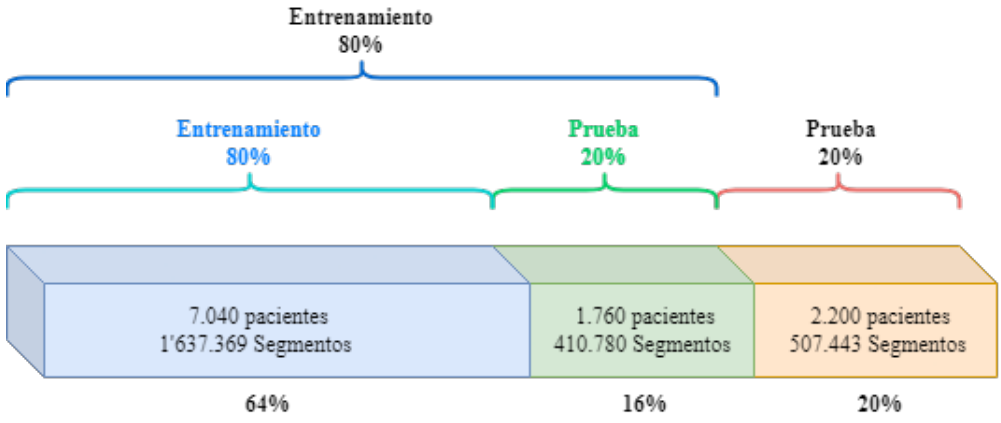
los datos brutos limitando las señales entre 5 [mV] y -5 [mV] (rango de señales cardíacas) (Tan et al., 2019). Las señales se registraron con una resolución de 16 bits y se muestrearon a 250 Hz.

2.3. Distribución y tratamiento de los datos

Nuestro modelo específico toma como punto de partida las señales etiquetadas que en numero de segmentos corresponden a 2.5 millones de señales. La distribución de la base de datos se realiza de forma distinta para los dos algoritmos propuestos. Para la evaluación de los algoritmos de compresión la distribución fue de 64 %, 16 % y 20 % para entrenamiento, validación y prueba respectivamente.

Figura 2

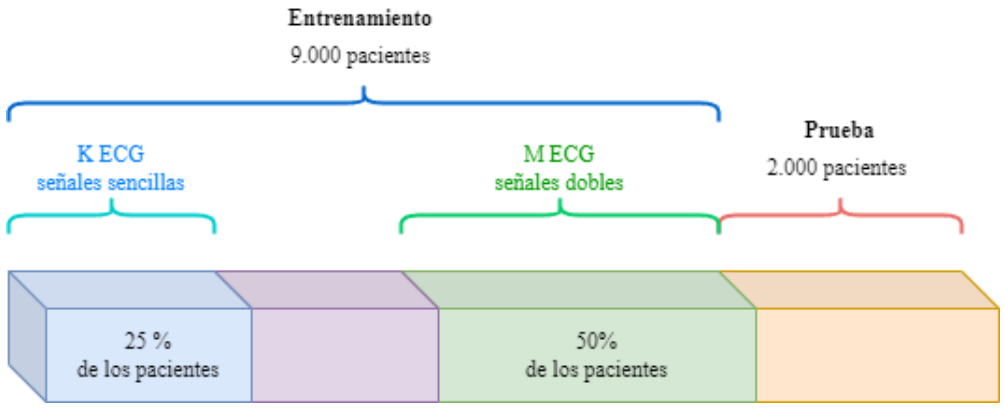
Distribución de la base de datos



En función del procedimiento de análisis contrastivo se requiere dividir la base de datos en dos sub procesos: pre-entrenamiento y entrenamiento. Por este motivo, empleamos dos longitudes diferentes de señales. Durante el proceso de entrenamiento, usamos K-señales que consisten en 2048 muestras, las cuales son llamadas datos individuales etiquetados. Por otro lado, en el proceso de pre-entrenamiento, nos basamos en M-datos no etiquetados dobles, que se refieren a señales compuestas por 4096 muestras. Esta variación en la longitud nos permite aplicar transformaciones temporales a los datos. Como se ilustra en la Figura 3 nos aseguramos de que los datos individuales y dobles provengan de conjuntos diferentes dentro del conjunto de datos de entrenamiento, con el fin de evitar el uso de los mismos pacientes en cada etapa.

Figura 3

Distribución de la base de datos



Es importante señalar que cada conjunto tiene pacientes separados; por lo tanto, no hay pacientes compartidos pacientes compartidos entre conjuntos.

2.4. Creación del Modelo Base

Como punto de partida se opta por el modelo de referencia presentado en (Hannun et al., 2019a; Rajpurkar et al., 2017) y se construye un modelo compacto a partir de él, reduciendo la profundidad y el ancho de la red manteniendo la mayor precisión posible. Este objetivo se basa en el hecho de que las redes neuronales a menudo están sobreparametrizadas, esto incrementa el

número de operaciones y memoria requeridas; reducirlas en profundidad disminuye estas cifras. La red presentada en estas dos investigaciones mostró un excelente desempeño en la detección de la fibrilación auricular, con un nivel superior al promedio de un cardiólogo (Rajpurkar et al., 2017).

La red consta de N bloques residuales, donde N es un hiperparámetro que nos permitirá cambiar la profundidad del modelo, en el modelo original es 15 (Rajpurkar et al., 2017). El modelo utiliza bloques residuales como una forma de resolver el problema de la explosión o decaimiento del gradiente (Glorot & Bengio, 2010). Cada bloque residual contiene capas convolucionales con un tamaño de kernel de 16. Además, el modelo incluye conexiones transitorias que contribuyen a la difusión de información en la red neuronal profunda. El modelo utiliza Batch normalization (Ioffe & Szegedy, 2015) para mantener los valores dentro de los límites y evitar la saturación. Se aplica Dropout (Srivastava et al., 2014) para evitar el sobreajuste durante los entrenamientos, como activación se usa la unidad de rectificación lineal (ReLU). Se ha demostrado que la función de activación de ReLU mejora tanto la precisión como el costo computacional por su fácil implementación en sistemas de hardware como FPGA, ya que se reduce a una lógica comparativa (Murat et al., 2020). A medida que la red se hace más profunda se aumenta el numero de canales de la siguiente forma:

$$C_{out}^{j} = I_{ch} \cdot 2^{s_j} \tag{2-1}$$

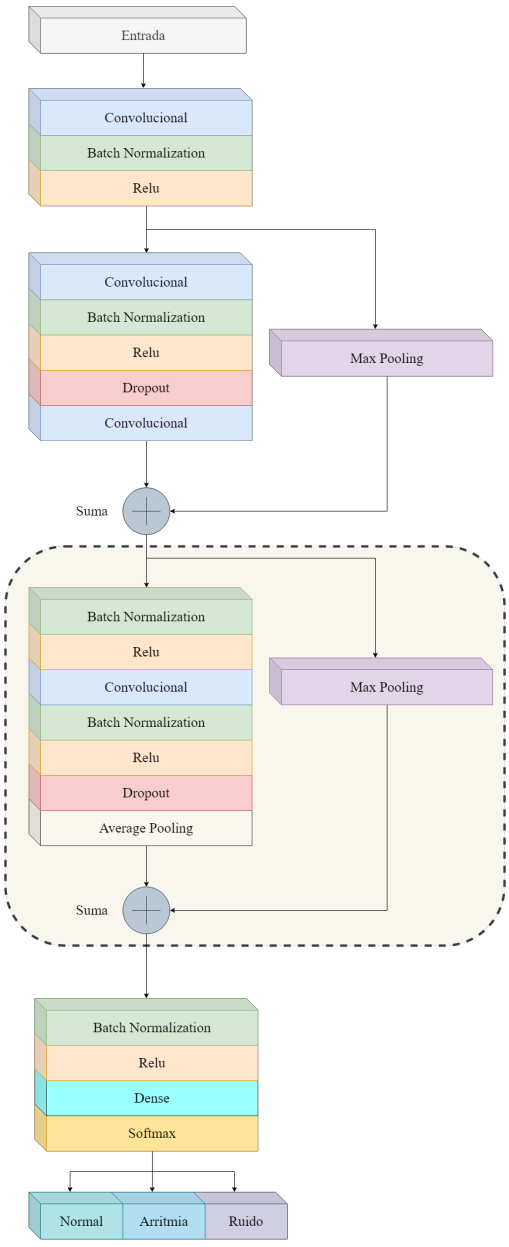
Donde C_{out}^{j} es el numero de canales de salida, I_{ch} es el numero de canales de la capa de entrada y 2^{s_j} es un factor de escalamiento que depende directamente del numero de bloques residuales. En el modelo original cada uno de los bloques residuales tiene un valor diferente de stride variando entre 1 y 2 alternamente. Adicionalmente cada bloque residual finaliza con una capa Average pooling; según el estado del arte, esta implementación ayuda a conservar las características principales de la entrada y en adición funciona como regularizador estructural que obliga explícitamente a que los mapas de características generalicen de forma confiable (Lin et al., 2013).

También se utilizaron capas de Maxpooling en las conexiones auxiliares para mantener la consistencia dimensional de los datos separados cuando se unían de nuevo en cada bloque. El

modelo termina con una capa totalmente conectada seguida de la función Softmax, que tiene 3 salidas correspondientes a la probabilidad de cada clase (Normal, Arritmia o Ruido). En la figura 4 se muestra la arquitectura básica de la red.

Figura 4

Arquitectura del modelo base



El modelo fue creado en el lenguaje de programación Python (Van Rossum & Drake, 2009)

desde un entorno alojado localmente, y para su descripción se usó la API de Keras (Chollet et al., 2015) y el framework seleccionado para el backend fué Tensorflow (Martín Abadi et al., 2015).

3. Estrategias de compresión

3.1. Compactación del modelo

El primer objetivo dentro del desarrollo de esta investigación fue la reducción del tamaño de la red, teniendo en cuenta tres aspectos: ancho, profundidad y número de parámetros; siendo este último aspecto casi una consecuencia de las dos primeras. Para lograrlo, se realizó una búsqueda del mínimo número de capas residuales, y el mínimo tamaño y número de filtros que permitían un buen desempeño de la red, de tal forma que conservara una exactitud similar al estado del arte. Esta investigación se centró en el número de parámetros entrenables de las capas convolucionales, ya que son la mayoría de los parámetros del modelo.

Sea C_{in}^{j} y C_{out}^{j} los tamaños de los canales de entrada y salida de cada capa convolucional 1D en el bloque j-residual (para simplificar la notación, suponemos que las activaciones de entrada y salida tienen el mismo tamaño espacial). La capa convolucional tendrá $C_{in}^{j} \cdot C_{out}^{j}$ filtros, por lo que, para un tamaño de núcleo de K, el número total de parámetros entrenables en la capa capa convolucional 1-D es,

$$P_{conv} = K \cdot C_{in}^{j} \cdot C_{out}^{j} + C_{out}^{j} \tag{3-1}$$

Donde C_{in}^{j} y C_{out}^{j} se calculan mediante la ecuación 2-1 ajustando el valor correspondiente de s_j . El último corresponde a los parámetros de sesgo Como se puede observar en la ecuación 3-1 el tamaño del filtro es directamente proporcional al número de parámetros y de la misma manera al número de operaciones.

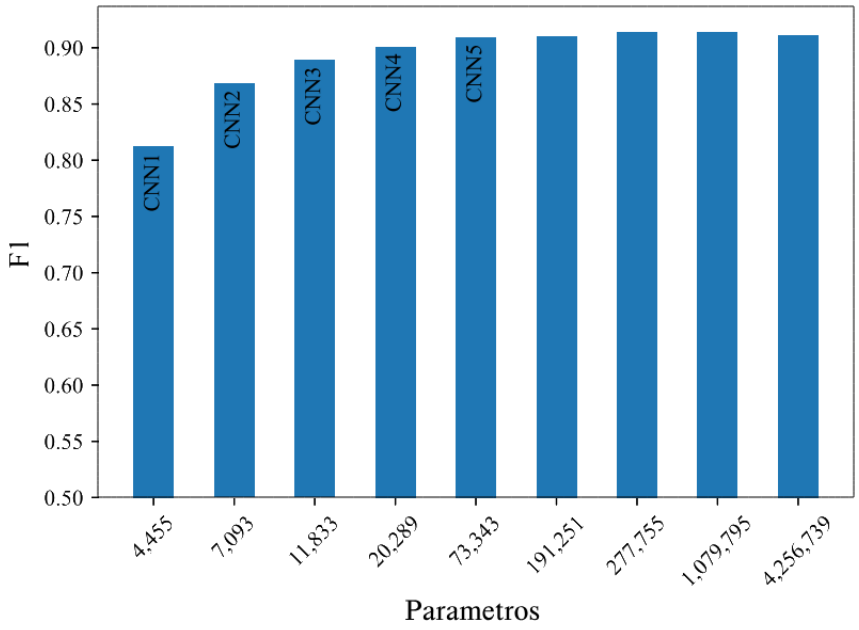
Se probaron varias configuraciones variando el tamaño del filtro, la profundidad y la anchura

de la red. También probamos diferentes formas de aplicar decimación en capas convolucionales estableciendo un paso superior a 1. Se utilizó la estrategia de decimación tardía, técnica que ha demostrado logros en el mejoramiento de la precisión con un presupuesto limitado de parámetros (Deng et al., 2020; K. He et al., 2015). Esta estrategia consigue mapas de activación más grandes al principio de la red, permitiendo una preservación de las características a lo largo de la inferencia y que a su vez, conduce a una mayor precisión en la clasificación (Iandola et al., 2016).

Se entrenaron un gran número de modelos con los cambios propuestos, variando los parámetros mencionados anteriormente buscando el mejor desempeño en F1 Score. De la figura 5 se puede ver como para un modelo con 73.343 parámetros ya se obtiene una puntuación F1 por arriba de 0.9. Adicionalmente, se puede apreciar que modelos más grandes no tienen una mejora significativa a pesar del incremento de parámetros.

Figura 5

F1 score para varios modelos



3.2. Compresión del modelo

Al observar que para grandes incrementos en el número de parámetros no hay mejoras considerables, se decidió continuar con los 5 modelos más pequeños, los cuales se presentan en la tabla 2 a continuación. Notese la variación de la cantidad de parámetros en términos de N y C_{out}^{j}.

Tabla 2

5 modelos con menos parámetros

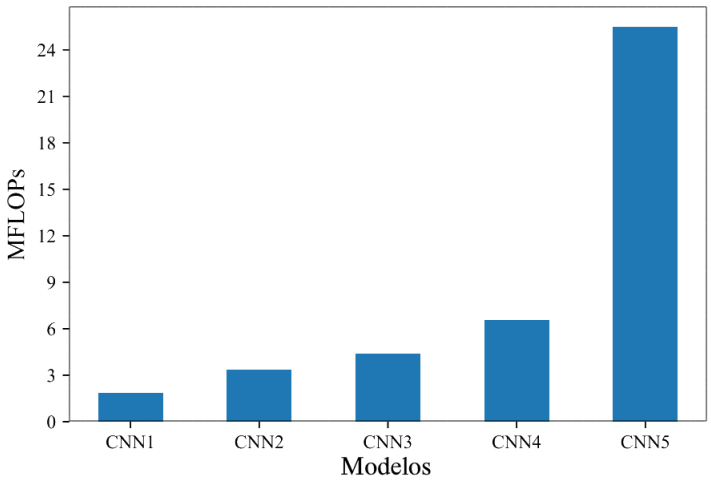
Modelo	Kernel size	N	I_{ch}	C_{out}^{j}	Parameters
CNN1	16	8	2	2,2,2,2,2,2,2,2	4455
CNN2	16	13	2	2,2,2,2,2,2,2,2,2,4,4,4,4,4	7093
CNN3	16	11	2	2,2,2,2,4,4,4,4,8	11833
CNN4	16	13	2	2,2,2,2,4,4,4,4,8,8,8,8,16	20289
CNN5	16	13	4	4,4,4,4,8,8,8,8,16,16,16,16,32	74343

Nota. Los modelos se ordenan por número de parámetros y se describen mediante tamaño de kernel, número de bloques, canales de entrada y canales de salida por bloque.

Para cada uno de estos modelos se calculó la cantidad de operaciones en punto flotante o FLOPs. Esta es una forma de determinar la complejidad computacional de los algoritmos y en este caso esta directamente relacionado con la reducción de parámetros, ya que menos parámetros implica menos operaciones. Como se puede observar en la figura 6 los modelos más pequeños tienen menor cantidad de FLOPs.

Figura 6

Cantidad de operaciones flotantes para cada uno de los 5 modelos seleccionados



A partir de estos modelos reducidos en parámetros se aplicaron las técnicas de destilación del conocimiento, la poda de pesos, la cuantización y las combinaciones entre estas técnicas para buscar el modelo más pequeño posible.

3.2.1. Destilación de conocimiento

Para la aplicación de esta técnica se realizaron varios experimentos, con la estrategia de búsqueda en malla, con el fin de obtener los parámetros más favorables α y temperatura. También se buscó que los modelos que funcionaron como *profesor* tuviesen como mucho 10 veces más parámetros que el modelo *estudiante*. Después de todos esos experimentos, se llegó a la conclusión que los valores de α debían permanecer en un rango menor a 0.5 y los valores de temperatura en valores menores a 15.

3.3. Poda

Este estudio aplicó el ”pruning”basado en la magnitud de los pesos, tanto a los modelos base como a los destilados. La técnica se aplicó a todas las capas del modelo utilizando dos formas

de poda:

- Anulando constantemente los pesos durante todo el proceso de entrenamiento
- Anulando gradualmente los pesos en una forma de decaimiento polinomial.

Los mejores resultados se obtuvieron al aplicar la primera forma. Los modelos CNN1-CNN5 fueron podados con un índice de dispersión del 50 % porque el uso de un índice mayor derivaba en un desempeño deficiente entre el puntaje F1 y el tamaño. No obstante, los modelos más grandes que CNN5 permitieron realizar la poda con un índice dedispersión superior al 50 % sin comprometer las métricas.

3.4. Cuantización

Esta investigación utilizó una implementación de cuantificación de 8 bits, basada en el trabajo de Jacob et al. (Jacob et al., 2018). Los autores presentan un esquema de cuantificación que cuantiza los pesos y las activaciones en enteros de 8 bits y algunos vectores de sesgo importantes en enteros de 32 bits, utilizando un Cuantizador Afín Uniforme. Además, proponen una multiplicación de matrices basada únicamente en aritmética entera, lo que permite una inferencia de 8 bits. Considerando una representación de 8 bits sin signo, mapeamos la variable de punto flotante a un rango de (0, 255). Para este propósito, se consideraron dos parámetros: Escala (S) y Punto cero (Z), que nos permiten mapear el valor de punto flotante (r) al valor correspondiente de 8 bits (q) mediante la siguiente ecuación:

$$r = S(q - Z) \tag{3-2}$$

El valor de punto cero es un número entero y el valor de escala es típicamente una cantidad en punto flotante. Cuando los pesos y las activaciones están siendo cuantizados, es necesario calcular correctamente los parámetros del cuantizador. Además, se requiere contar con algunos datos de calibración para calcular los rangos dinámicos de las activaciones. Ahora, con los parámetros

del cuantizador, es posible encontrar los valores adecuados de escala y punto cero para realizar la cuantización (Ecuación 6). Finalmente, en el momento de la inferencia, debemos convertir la imagen de entrada en valores enteros de 8 bits.

4. Análisis contrastivo

Para nuestro proceso, empleamos como codificador el modelo presentado en la Figura 4. Este modelo nos brinda la flexibilidad de ajustar el tamaño del modelo al modificar las dimensiones del filtro convolucional inicial. Es importante señalar que el codificador no contiene la capa de clasificación. Asimismo, creamos una capa de proyección según lo descrito en SimCLR (Chen et al., 2020), cuyo tamaño de entrada depende de la longitud de la salida del codificador. Esta red neuronal pequeña está compuesta por dos capas densas con un número adecuado de unidades, definido como el ancho. Por otro lado, diseñamos una capa de sondeo lineal (linear probe) para llevar a cabo la tarea de clasificación. Las capas finales se componen de una capa densa con 128 neuronas y otra capa densa con 3 neuronas (capa de clasificación).

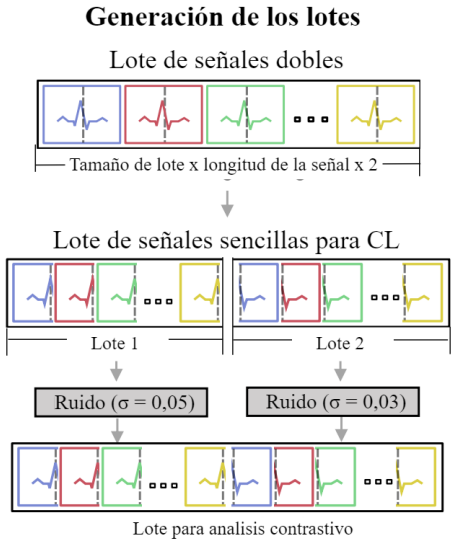
4.1. Transformaciones

En este estudio, basado en los trabajos de Chen et al. (Chen et al., 2020), Kiyasseh et al. (Kiyasseh et al., 2021) y Krizhevsky et al. (Krizhevsky et al., 2012), se realiza la Ampliación de Datos mediante la subdivisión temporal (dividiendo en dos) y la aplicación de ruido gaussiano (con desviaciones estándar diferentes para cada transformación), tal como se muestra en la Figura 7. Se probaron otras transformaciones pero no se obtuvieron resultados favorables.

Dado que las señales icentia11k tienen una dimensionalidad específica (un solo lead), no es posible aplicar la Ampliación de Datos en múltiples leads. Por esta razón, es necesario utilizar datos con el doble de ejemplos de entrenamiento para el pre-entrenamiento, lo cual permite llevar a cabo transformaciones temporales. Se crea el lote de aprendizaje contrastivo de acuerdo a lo ilustrado en la Figura 7.

Figura 7

Metodología para el aumento de datos para el análisis contrastivo



4.2. Experimentación

Realizamos nuestros experimentos en dos fases principales, como se representa en la Figura 7. En el proceso de pre-entrenamiento, incorporamos dos capas densas, cada una con 128 neuronas. Estas capas se emplean para la proyección. Nuestro enfoque implica el uso de una tasa de aprendizaje programada que comienza en 0.01, junto con la implementación de interrupciones tempranas en el entrenamiento. Iniciamos el proceso pre-entrenando el codificador mediante el uso del aprendizaje contrastivo. Posteriormente, aplicamos una tarea de clasificación para evaluar la inicialización que ha sido introducida mediante el pre-entrenamiento.

La secuencia de este procedimiento se detalla de la siguiente manera:

- Seleccionamos M-señales de los datos no etiquetados en su forma doble para construir lotes contrastivos de un tamaño específico, tal como se ilustra en la Figura 7

- Utilizamos los lotes contrastivos para llevar a cabo el pre-entrenamiento del codificador, empleando las capas de proyección. Siguiendo la metodología de Chen et al. (Chen et al.,

2020), aplicamos la pérdida de entropía cruzada normalizada con una temperatura escalada, donde $\tau = 0{,}1$ actúa como el parámetro de temperatura. La función de pérdida tiene la finalidad de maximizar la similitud entre las proyecciones de pares positivos. En la ecuación 4.2 se exhibe la función de pérdida para un par positivo a i, j de señales, en donde z hace referencia a las salidas de las capas de proyección, y N representa el tamaño del lote contrastivo (Kiyasseh et al., 2021). La pérdida definitiva se calcula para los N pares positivos en cada lote contrastivo de la siguiente forma:

$$l_{i,j} = -\log \frac{\exp\left[sim(zi, zj)/\tau\right]}{\sum_{s=1}^{2N, s\neq i} \exp\left[sim(zi, zs)/\tau\right]} \tag{4-1}$$

Aquí, la función **sim(x)** corresponde a la función de la similaridad del coseno.

- Las métricas contrastivas que empleamos son la pérdida contrastiva y la precisión.

Para el proceso de entrenamiento se emplea la función de pérdida Sparse Categorical Crossentropy y el optimizador Adam con los parámetros estándar. Utilizamos una tasa de aprendizaje programada que comienza en 0.001, y también aplicamos llamadas de detención temprana durante el entrenamiento. Realizamos este proceso de la siguiente manera:

- Una vez que se ha concluido la etapa de pre-entrenamiento, tomamos el codificador pre-entrenado y le agregamos la capa de sondeo lineal. Finalmente, entrenamos el codificador utilizando K-señales de los datos individuales, como se muestra en la Figura 7.

- Las métricas de clasificación que evaluamos son la pérdida y el puntaje F1.

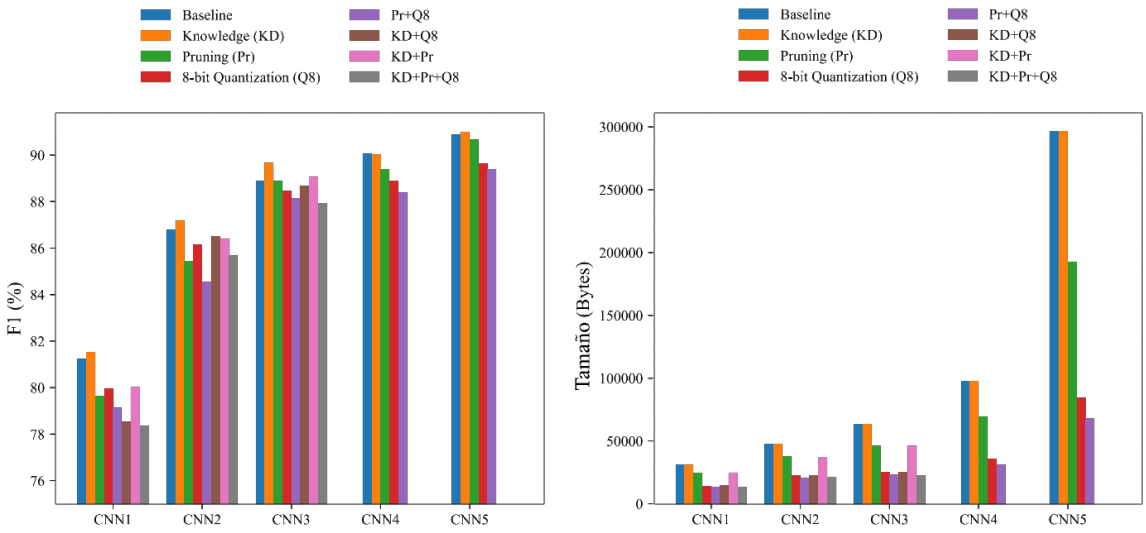
5. Análisis y Resultados

5.1. Análisis

En cuanto a la compresión de modelos, se probaron diversas combinaciones de técnicas, en búsqueda de la mejor relación de compresión y exactitud. Realizamos las simulaciones en GPUs Tesla V100 SXM2. La Figura 8 muestra los resultados relacionados de F1 y tamaño para las 5 redes seleccionadas previamente, para las diferentes pruebas y combinaciones.

Figura 8

Exactitud F1 y tamaño en Bytes de los modelos CNN1-CNN5 usando técnicas de compresión



(a) 5k datos etiquetados.

(b) 10k datos etiquetados.

Nota. La subfigura **(a)** presenta la exactitud F1 y la subfigura **(b)** presenta el tamaño en Bytes para los modelos CNN1-CNN5 de la Tabla 2, usando Destilación de conocimiento (KD), Pruning (Pr), Cuantización 8-bit (Q8) y sus combinaciones.

Se obtuvieron mejores resultados utilizando un esquema de ”pruningçonstante. Los modelos CNN1-CNN5 fueron ”pruneadosçon una esparcidad del 50 %, ya que una esparcidad mayor resultó en un mal equilibrio entre el puntaje F1 y el tamaño. Sin embargo, los modelos más grandes que CNN5 permitieron un ”pruningçon una esparcidad mayor al 50 % sin comprometer las métricas. Los resultados del ”pruning”se muestran en la Figura 8. Se presentan los modelos base ”pruneados”(verde) y los modelos destilados ”pruneados”(rosa).

Al observar la imagen se puede evidenciar que en los casos que se aplico destilación de conocimiento se incrementó la exactitud. Así mismo se puede observar como el uso de KD + Pr en los 3 primeros modelos, logra una reducción adicional del tamaño sin comprometer significativamente (más del 1 %) la precisión en comparación con el modelo base, siendo el modelo CNN3 el que representa mejor esta relación.

Por otra parte, en el análisis contrastivo también entrenamos numerosos modelos y calculamos el puntaje F1, que es la media armónica de la precisión y la exhaustividad. Además, utilizamos una estratificación de datos con 10 divisiones para calcular la media y la desviación estándar de los resultados. Realizamos las simulaciones en GPUs Tesla V100 SXM2. Tanto para el proceso de pre-entrenamiento como para el entrenamiento, establecemos un tamaño de lote de 256 señales de ECG.

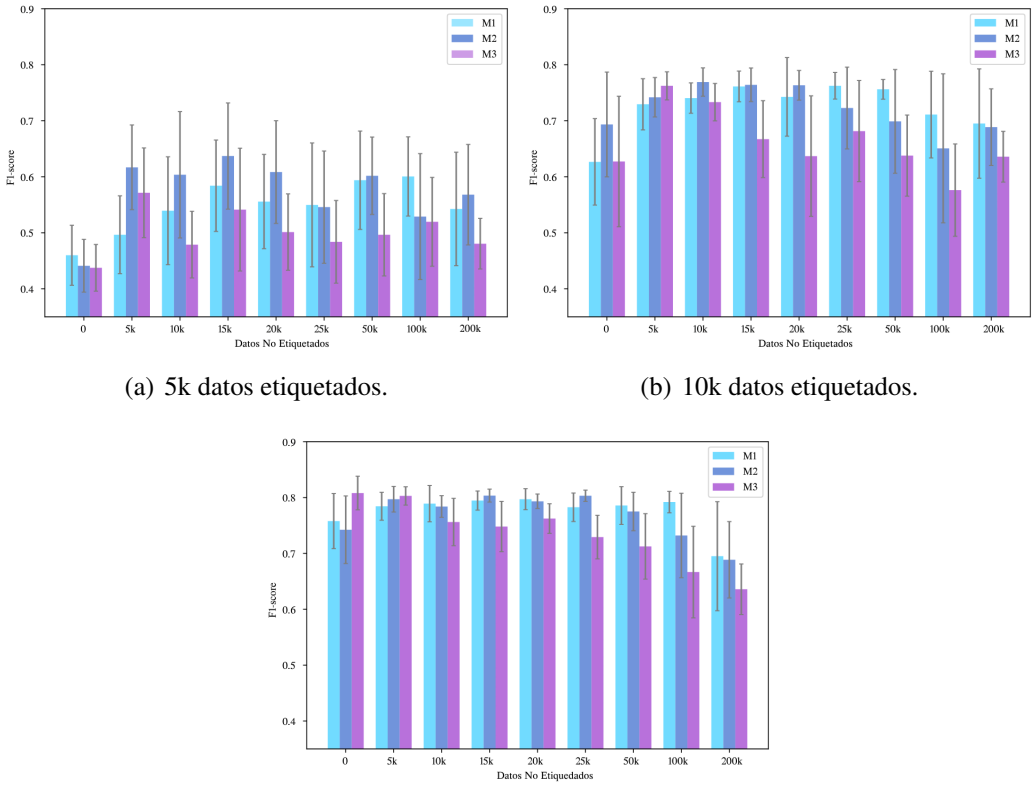
La Figura 9 muestra los resultados relacionados con la cantidad de datos etiquetados, datos no etiquetados y el tamaño del modelo. Los datos etiquetados varían entre 5k, 10k y 15k señales de ECG, como se presenta en las subfiguras de la Figura 9. También utilizamos tres modelos con diferentes cantidades de parámetros: M1 (naranja), M2 (azul) y M3 (rojo). Estos valores corresponden a 66k, 250k y 1M de parámetros. Los datos no etiquetados varían entre 0, 20k, 25k, 50k, 100k y 200k. El valor cero indica que el modelo tiene una inicialización aleatoria, es decir, no hay un proceso de pre-entrenamiento.

En la Figura 9, se observa una mejora significativa en el puntaje F1 promedio al utilizar el Aprendizaje Contrastivo (CL) en comparación con la inicialización aleatoria con pocos datos etiquetados. Además, nuestros resultados sugieren que 15k de señales de ECG no etiquetados

pueden lograr mejores resultados.

Figura 9

F-1 Score para diferentes cantidades de datos no etiquetados



(a) 5k datos etiquetados.

(b) 10k datos etiquetados.

(c) 15k datos etiquetados.

5.2. Discusión

Se realiza una comparación entre nuestro método y los métodos de CLOCS (Kiyasseh et al., 2021) mostrado en la Tabla 3. Observamos que nuestro trabajo presenta una mejora en comparación con CLOCS en condiciones cercanas. No obstante, reconocemos que las desviaciones estándar son altas en la mayoría de nuestros resultados. Esta alta desviación podría deberse al uso de un conjunto de pruebas más grande con una cantidad significativa de señales ruidosas. Los datos unidimensionales han demostrado ser un desafío en el contexto del Aprendizaje Contrastivo (CL). Este tipo de datos reduce el número de posibilidades en la Ampliación de Datos (DA), lo cual

Tabla 3

Tablas comparativas sobre la mejora con diferentes cantidades de datos etiquetados y conjuntos de datos

Método	Datos de entrenamiento	Datos etiquetados	Datos de prueba	Inicialización aleatoria	Inicialización Contrastiva	AUC media mejora
CLOCS	Cardiology (Hannun et al., 2019b)	~3k	~1k	0.669 ± 0.007	0.708 ± 0.017	5.8 %
	Physionet 2017 (Clifford et al., 2017)	~5k	~6k	0.738 ± 0.014	0.770 ± 0.012	4.6 %
	Physionet 2020 (Alday et al., 2020)	~16k	~16k	0.766 ± 0.005	0.801 ± 0.013	4.3 %
Ours	Icentia 11k (Tan et al., 2019)	~5k	~465k	0.459 ± 0.053	0.601 ± 0.071	30.5 %
		~10k		0.627 ± 0.077	0.763 ± 0.024	21.6 %
		~15k		0.742 ± 0.060	0.803 ± 0.010	8.1 %

Nota. Los valores reportan la media y desviación estándar de los resultados. AUC corresponde al área bajo la curva y CL al aprendizaje contrastivo.

es importante para mejorar el aprendizaje en la etapa de pre-entrenamiento. Nuestros resultados sugieren que aumentar la cantidad de datos etiquetados o el tamaño del modelo disminuye el impacto de nuestro método. Por otro lado, también indican que la cantidad de datos no etiquetados utilizados en el pre-entrenamiento determina el éxito de la técnica de implementación. Encontrar la cantidad óptima de datos no etiquetados es necesario para obtener el mejor rendimiento del aprendizaje contrastivo.

La Tabla 4 realiza una comparación entre algunos estudios recientes de clasificación de ECG y nuestros modelos propuestos. Hemos incluido únicamente aquellos estudios de investigación que han proporcionado información sobre la cantidad de parámetros utilizados en sus modelos. Es importante destacar que la cantidad de parámetros necesarios aumenta cuando se trabaja con un mayor número de pacientes, lo que a su vez mejora la capacidad de generalización. En nuestro caso, hemos empleado el conjunto de datos públicos de ECG más grande tanto en términos de la cantidad de señales como de pacientes. Nuestro modelo CNN-4 (Baseline), con tan solo 20,289 parámetros, logra obtener un puntaje F1 de 0.9, estableciendo así un nuevo récord de precisión en conjuntos de datos con más de 9,000 pacientes.

Tabla 4

Recent studies classifying ECG signals using Deep Learning techniques

Estudio	Base de datos	Datos totales & Clases	Numero de pacientes	Técnicas	Numero de parámetros	Resultados
Hannun et al., 2019 (Hannun et al., 2019a)	Zio Monitor	91,232 señales 12 clases	53,549 pacientes	CNN	10,473,635	F1 = 0.837
Rubin et al., 2018 (Rubin et al., 2018)	PhysioNet Challenge 2017	8,528 señales 4 clases	8,528 pacientes	CNN	262,344	F1 = 0.820
Andersen et al., 2019 (Andersen et al., 2019)	MIT-BIH AD	23 long-term señales 2 clases	47 pacientes	CNN-LSTM	159,841	Acc = 0.978 Se = 0.989 Sp = 0.969
Castillo et al., 2020 (Castillo et al., 2020)	MIT-BIH AD	412,152 señales 2 clases	47 pacientes	CNN	12,175	Acc = 0.974 Se = 0.970 Sp = 0.978
Fan et al., 2020 (Fan et al., 2020)	MIT-BIH AD CINC Challenge Dataset	8,249 señales 3 clases	47 pacientes	CNN	13,024,821	Balanced set: F1 = 0.840 Acc = 0.850 Imbalanced set: F1 = 0.850 Acc = 0.870
Guo et al., 2019 (Guo et al., 2019)	MIT-BIH AD MIT-BIH Supraventricular AD	289,666 latidos 5 clases	47 pacientes	CNN	18,000,000	VEB: Acc = 0.937 Se = 0.912 Sp = 0.947 SVEB: Acc = 0.936 Se = 0.627 Sp = 0.964
Oh et al., 2019 (Oh et al., 2018)	MIT-BIH AD	94,667 latidos 5 clases	47 pacientes	Mofified U-Net	24,162	Acc = 0.9732
Yildirim et al., 2019 (Yildirim et al., 2019)	MIT-BIH AD	100,022 latidos 5 clases	47 pacientes	16-L dCAE 5-L LSTM	560,121	Raw ECG: Acc = 0.992 Coded Features: Acc = 0.991
Gopika et al., 2020 (Gopika et al., 2020)	Kaggle Arrhythmia ECG heartbeat	109,466 latidos 5 clases	47 pacientes	RCNN	-	Pr = 0.980 Re = 0.980 F1 = 0.980
Yao et al., 2020 (Yao et al., 2020)	1st China Physiological Signal Challenge	9,831 señales 8 clases	9,831 pacientes	ATI-CNN	4,984,640	Acc = 0.812 Pr = 0.826 Re = 0.801 F1 = 0.812
CNN1 (KD+Pr) (**Nuestra**)	Icentia11k	2,555,592 señales 4 clases	11,000 pacientes	CNN	2,324	Acc = 0.881 Pr = 0.790 Re = 0.820 F1 = 0.800
CNN3 (KD+Pr) (**Nuestra**)	Icentia11k	2,555,592 señales 4 clases	11,000 pacientes	CNN	6,173	Acc = 0.924 Pr = 0.884 Re = 0.898 F1 = 0.890
CNN4 (Modelo Base) (**Nuestra**)	Icentia11k	2,555,592 señales 4 clases	11,000 pacientes	CNN	20,289	Acc = 0.929 Pr = 0.903 Re = 0.898 F1 = 0.900

Nota. AD: Arrhythmia Database; VEB: V class versus [N, S, and F]; SVEB: S class versus [N, V and F]; Acc: accuracy; Pr: precision; Re: recall; Se: sensitivity; Sp: specificity; F1: F1-score; ATI-CNN: attention-based time-incremental CNN; dCAE: deep convolutional auto-encoder.

6. Conclusiones y recomendaciones

6.1. Conclusiones

Esta investigación abordó los retos relacionados con la detección de Fibrilación Auricular en dispositivos portátiles desde una óptica de compresión del modelo y el procesamiento de los datos. Propusimos varios modelos de aprendizaje profundo cuya complejidad computacional era lo suficientemente baja como para ser desplegados en dispositivos portátiles. Podemos afirmar que es posible y confiable la realización de modelos comprimidos para la detección de arritmias cardiacas como la Fibrilación Auricular sin comprometer significativamente la precisión del modelo.

Por otra parte, el enfoque centrado en el procesamiento de los datos a través de la técnica del análisis contrastivo nos mostró que la técnica mejora la media F1 en modelos pequeños con pocos datos etiquetados, lo cual resulta mucho mejor para desplegar dispositivos portátiles. Sin embargo, los modelos más grandes y con muchos datos etiquetados no presentan las mejores condiciones para nuestro método.

Todo lo anterior nos da una oportunidad de desarrollo mas profundo, en consecuencia, la investigación futura debe centrarse en la prueba de concepto de un dispositivo portátil y la evaluación preclínica. Se espera que con los resultados de este estudio se pueda contribuir a la detección precoz de la FA en tiempo real como apoyo al diagnostico médico, diagnóstico que es crucial para prescribir un tratamiento oportuno, reduciendo así el riesgo de morbilidad y mortalidad.

Referencias Bibliográficas

Acharya, U. R., Fujita, H., Lih, O. S., Hagiwara, Y., Tan, J. H., & Adam, M. (2017). Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network. *Information sciences*, *405*, 81-90.

Alday, E. A. P., Gu, A., Shah, A. J., Robichaux, C., Wong, A.-K. I., Liu, C., Liu, F., Rad, A. B., Elola, A., Seyedi, S., et al. (2020). Classification of 12-lead ecgs: the physionet/computing in cardiology challenge 2020. *Physiological measurement*, *41*(12), 124003.

Andersen, R. S., Peimankar, A., & Puthusserypady, S. (2019). A deep learning approach for real-time detection of atrial fibrillation. *Expert Systems with Applications*, *115*, 465-473.

Anwar, S., Hwang, K., & Sung, W. (2017). Structured Pruning of Deep Convolutional Neural Networks. *J. Emerg. Technol. Comput. Syst.*, *13*(3). https://doi.org/10.1145/3005348

Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep? *Advances in neural information processing systems*, *27*.

Bhatt, H. V., & Fischer, G. W. (2015). Atrial fibrillation: pathophysiology and therapeutic options. *Journal of Cardiothoracic and Vascular Anesthesia*, *29*(5), 1333-1340.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning? *Proceedings of machine learning and systems*, *2*, 129-146.

Boulch, A. (2017). Sharesnet: reducing residual network parameter number by sharing weights. *arXiv preprint arXiv:1702.08782*.

Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 535-541.

Castillo, J. A., Granados, Y. C., & Fajardo, C. A. (2020). Patient-specific detection of atrial fibrillation in segments of ECG signals using deep neural networks. *Ciencia E Ingenieria Neogranadina*, *30*(1), 45-58.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020-18 de julio). A Simple Framework for Contrastive Learning of Visual Representations. En H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning* (pp. 1597-1607, Vol. 119). PMLR. https://proceedings.mlr.press/v119/chen20j.html

Chollet, F., et al. (2015). Keras. https://github.com/fchollet/keras

Čihák, R., Haman, L., & Táborskỳ, M. (2016). 2016 ESC Guidelines for the management of atrial fibrillation developed in collaboration with EACTS. *Cor et vasa*, *6*(58), e636-e683.

Clifford, G. D., Liu, C., Moody, B., Lehman, L. H., Silva, I., Li, Q., Johnson, A. E., & Mark, R. G. (2017). AF classification from a short single lead ECG recording: The PhysioNet/computing in cardiology challenge 2017. *Computing in Cardiology*, *44*, 1-4. https://doi.org/10.22489/CinC.2017.065-469

Datta, S., Puri, C., Mukherjee, A., Banerjee, R., Choudhury, A. D., Singh, R., Ukil, A., Bandyopadhyay, S., Pal, A., & Khandelwal, S. (2017). Identifying normal, AF and other abnormal ECG rhythms using a cascaded binary classifier. *2017 Computing in cardiology (cinc)*, 1-4.

Deng, B. L., Li, G., Han, S., Shi, L., & Xie, Y. (2020). Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey. *Proceedings of the IEEE*, *108*, 485-532.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., & Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. *Artificial intelligence and statistics*, 153-160.

Fan, X., Hu, Z., Wang, R., Yin, L., Li, Y., & Cai, Y. (2020). A novel hybrid network of fusing rhythmic and morphological features for atrial fibrillation detection on mobile ECG signals. *Neural Computing and Applications*, *32*(12), 8101-8113.

Forero-Gómez, J. E., Moreno, J. M., Agudelo, C. A., Rodríguez-Arias, E. A., & Sánchez-Moscoso, P. A. (2017). Fibrilación auricular: enfoque para el médico no cardiólogo. *Iatreia*, *30*(4), Pág. 404-422. https://doi.org/10.17533/udea.iatreia.v30n4a05

Galvez-Olortegui, J. K., Álvarez-Vargas, M. L., Galvez-Olortegui, T. V., Godoy-Palomino, A., & Camacho-Saavedra, L. (2016). Current clinical practice guidelines in atrial fibrillation: a review. *Medwave*, *16*(01).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249-256.

Gopika, P., Sowmya, V., Gopalakrishnan, E., & Soman, K. (2020). Transferable approach for cardiac disease classification using deep learning. En *Deep Learning Techniques for Biomedical and Health Informatics* (pp. 285-303). Elsevier.

Guo, L., Sim, G., & Matuszewski, B. (2019). Inter-patient ECG classification with convolutional and recurrent neural networks. *Biocybernetics and Biomedical Engineering*, *39*(3), 868-879.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, *28*.

Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019a). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, *25*(1), 65-69.

Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019b). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. https://doi.org/10.1038/s41591-018-0268-3

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026-1034. https://doi.org/10.1109/ICCV.2015.123

He, Y., Zhang, X., & Sun, J. (2017). Channel Pruning for Accelerating Very Deep Neural Networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Hinton, G., Vinyals, O., Dean, J., et al. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, *2*(7).

Hong, S., Wu, M., Zhou, Y., Wang, Q., Shang, J., Li, H., & Xie, J. (2017). ENCASE: An ENsemble ClASsifiEr for ECG classification using expert features and deep neural networks. *2017 Computing in cardiology (cinc)*, 1-4.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448-456.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704-2713.

Kiyasseh, D., Zhu, T., & Clifton, D. A. (2021-24 de julio). CLOCS: Contrastive Learning of Cardiac Signals Across Space, Time, and Patients. En M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 5606-5615, Vol. 139). PMLR. https://proceedings.mlr.press/v139/kiyasseh21a.html

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Mahajan, R., Kamaleswaran, R., Howe, J. A., & Akbilgic, O. (2017). Cardiac rhythm classification from a short single lead ECG recording via random forest. *2017 Computing in Cardiology (CinC)*, 1-4.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, . . . Xiaoqiang Zheng. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (1st ed.) [Software available from tensorflow.org. Disponible en: https://www.tensorflow.org/]. Google Brain. Mountain View.

Mirzadeh, S. I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., & Ghasemzadeh, H. (2020). Improved knowledge distillation via teacher assistant. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*, 5191-5198.

Mozer, M. C., & Smolensky, P. (1988). Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, *1*.

Murat, F., Yildirim, O., Talo, M., Baloglu, U. B., Demir, Y., & Acharya, U. R. (2020). Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review. *Computers in Biology and Medicine*, *120*, 103726. https://doi.org/https://doi.org/10.1016/j.compbiomed.2020.103726

Oh, S. L., Ng, E. Y., Tan, R. S., & Acharya, U. R. (2018). Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats. *Computers in Biology and Medicine*, *102*(April), 278-287. https://doi.org/10.1016/j.compbiomed.2018.06.002

Paquet, P., Levesque, D., & Fecteau, P. (2022, 19 de abril). *Adhesive extender for medical electrode and use thereof with wearable monitor* (1st ed.) [US Patent 11,304,660]. Google Patents. United States.

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. (2017). Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*.

Romero, M., & Chávez, D. (2014). Carga de enfermedad atribuible a fibrilación auricular en Colombia (2000-2009). *Revista Colombiana de Cardiología*, *21*(6), 374-381.

Rubin, J., Parvaneh, S., Rahman, A., Conroy, B., & Babaeizadeh, S. (2018). Densely connected convolutional networks for detection of atrial fibrillation from short single-lead ECG recordings. *Journal of electrocardiology*, *51*(6), S18-S21.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818-2826.

Tan, S., Androz, G., Chamseddine, A., Fecteau, P., Courville, A., Bengio, Y., & Cohen, J. P. (2019). Icentia11k: An unsupervised representation learning dataset for arrhythmia subtype discovery. *arXiv preprint arXiv:1910.09570*.

Teijeiro, T., García, C. A., Castro, D., & Félix, P. (2017). Arrhythmia classification from the abductive interpretation of short single-lead ECG records. *2017 Computing in cardiology (cinc)*, 1-4.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual* (1st ed.). CreateSpace.

Xu, S., Huang, A., Chen, L., & Zhang, B. (2020). Convolutional Neural Network Pruning: A Survey. *2020 39th Chinese Control Conference (CCC)*, 7458-7463. https://doi.org/10.23919/CCC50068.2020.9189610

Yao, Q., Wang, R., Fan, X., Liu, J., & Li, Y. (2020). Multi-class Arrhythmia detection from 12-lead varied-length ECG using Attention-based Time-Incremental Convolutional Neural Network. *Information Fusion*, *53*(June 2019), 174-182.

Yildirim, O., Baloglu, U. B., Tan, R.-S., Ciaccio, E. J., & Acharya, U. R. (2019). A new approach for arrhythmia classification using deep coded features and LSTM networks. *Computer methods and programs in biomedicine*, *176*, 121-133.

Yuan, L., Tay, F. E., Li, G., Wang, T., & Feng, J. (2019). Revisit knowledge distillation: a teacher-free framework.

Zabihi, M., Rad, A. B., Katsaggelos, A. K., Kiranyaz, S., Narkilahti, S., & Gabbouj, M. (2017). Detection of atrial fibrillation in ECG hand-held devices using a random forest classifier. *2017 Computing in Cardiology (CinC)*, 1-4.

Zhai, M., Chen, L., Tung, F., He, J., Nawhal, M., & Mori, G. (2019). Lifelong gan: Continual learning for conditional image generation. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2759-2768.

Zhang, Y., Xiang, T., Hospedales, T. M., & Lu, H. (2018). Deep mutual learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4320-4328.