UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Clasificación automática de latidos de un electrocardiograma utilizando aprendizaje profundo

Por:
Carlos Andrés Villagrán Fuentes

Profesor Guía:
Dr. Lorena Pradenas Rojas

Concepción, Abril de 2017

Tesis presentada a la

DIRECCIÓN DE POSTGRADO

UNIVERSIDAD DE CONCEPCIÓN

Para optar al grado de

MAGÍSTER EN INGENIERÍA INDUSTRIAL

AGRADECIMIENTOS

Esta tesis fue desarrollada con el apoyo de la beca CONICYT magíster nacional año 2015 (CONICYT-PCHA/MagísterNacional/2015-22152206).

RESUMEN

CLASIFICACIÓN AUTOMÁTICA DE LATIDOS DE UN ELECTROCARDIOGRAMA UTILIZANDO APRENDIZAJE PROFUNDO

"Carlos Andrés Villagrán Fuentes"

Abril de 2017

PROFESOR GUÍA: "Lorena Pradenas Rojas"

PROGRAMA: Magíster en Ingeniería Industrial

Las enfermedades cardiovasculares corresponden a la primera causa de muerte en el mundo y conllevan un elevado costo. Además, la fuerza de trabajo de cardiólogos va en constante decrecimiento. Por ello, se han desarrollado algoritmos automáticos de clasificación de latidos cardíacos desde registros de electrocardiograma (ECG).

Este trabajo abordó esta problemática mediante aprendizaje profundo. Se utilizó la base de datos de arritmias del MIT-BIH, que consta de 48 registros de 30 min con 2 canales de ECG y que posee las etiquetas de cada latido. Se realizó un procesamiento de los datos, se conformaron conjuntos de entrenamiento y prueba bajo un enfoque inter-paciente, y se implementaron 4 arquitecturas de redes neuronales convolucionales con distintas profundidades, utilizando Python 2.7 y "Tensorflow".

Se lograron exactitudes de 93,2%; 93,7%; 94,4% y 94,0% para las 4 redes implementadas, respectivamente; lo que posiciona a este trabajo como un aporte a la literatura actualmente disponible para resolver el problema.

Palabras claves: Electrocardiograma, Aprendizaje de Máquina, Clasificación, Red Neuronal Convolucional

ABSTRACT

AUTOMATIC HEARTBEAT CLASSIFICATION FROM ELECTROCARDIOGRAM USING DEEP LEARNING

"Carlos Andrés Villagrán Fuentes"

April, 2017

THESIS SUPERVISOR: "Lorena Pradenas Rojas"

PROGRAM: Master in Industrial Engineering

Cardiovascular diseases are the leading cause of death in the world and they carry a high cost. In addition, cardiologists are not always available at all times or in all places. Therefore, several algorithms for automatic classification of heartbeats from electrocardiogram (ECG) have been developed.

In this work, the problem was addressed through deep learning techniques. The MIT-BIH arrhythmia database was used, which consists of 48 records of 30 min with 2 ECG channels that have the labels of each heartbeat. Data processing was performed, and sets for test and training under an inter-patient approach were constructed. The 5 super-classes from AAMI standard were considered. Four conventional neural network architectures with different depths were CPU-implemented using Python 2.7 and "Tensorflow."

Accuracies of 93.2%, 93.7%, 94.4% and 94.0% were achieved for the 4 networks implemented, which indicates that this study contributes to the literature currently available to solve the problem.

Palabras claves: Electrocardiogram, Machine Learning, Classification, Convolutional Neural Network

TABLA DE CONTENIDOS

LISTADO DE TABLAS

LISTADO DE FIGURAS

Capítulo 1. Introducción

Según la organización mundial de la salud (OMS), las enfermedades cardiovasculares (ECV) están posicionadas como la principal causa de muerte en el mundo, siendo responsables del 31% del total de muertes, con 17,5 millones de personas en 2012. Las afecciones cardíacas consideradas corresponden a: cardiopatías coronarias, enfermedades cerebrovasculares, arteriopatías periféricas, cardiopatías reumáticas, cardiopatías congénitas, trombosis venosas profundas y embolias pulmonares. Las recomendaciones y planes estratégicos de la OMS se enfocan en la prevención y detección temprana de estas enfermedades, a través de la modificación de factores de riesgo (fisiológicos y conductuales) y de la disponibilidad de tecnología y medicamentos [1].

Las ECV, así como también otras enfermedades no transmisibles, conllevan un elevado gasto para la sociedad. En el 2011, el foro económico mundial (WEF) desarrolló un modelo de costos para estimar la carga económica de las ECV utilizando la información disponible de la OMS y considerando: detección, prevención primaria, prevención secundaria, atención hospitalaria aguda y pérdida en productividad. Los resultados muestran que para el 2030 el costo global de las ECV alcanzaría los US$1.044 miles de millones y que se gastaría alrededor de US$20 billones en el periodo de 20 años entre 2010 y 2030 [2].

Muchas de las ECV están relacionadas a condiciones o patologías en el sistema de generación y conducción eléctrico del corazón y se denominan arritmias cardíacas [3]. Vásquez et al. [4] analizaron la frecuencia de arritmias cardíacas y trastornos de la conducción en una consulta de cardiología general en un hospital de España, llegando a la conclusión que estas enfermedades se encuentran presentes en un 39% de los pacientes atendidos.

Las arritmias se pueden clasificar en [5]:

I. **Bradicardias:** Frecuencia cardíaca inferior a 60 latidos por minuto con ritmo regular (bradicardia) o irregular (bradiarritmia). Algunos tipos de bradicardias son: bradicardia sinusal, arritmia sinusal respiratoria, síndrome del nódulo sinoauricular enfermo, bloqueos sinoauriculares y parada sinusal, síndrome del seno carotídeo, bloqueos a-v,

bloqueos de rama, bloqueos bi y trifasiculares, y ritmos de reemplazo.

II. **Taquicardias:** Frec. cardíaca sobre 100 latidos por minuto. Se dividen en:

   a) ***Taquiarritmias supreventriculares:*** Se originan en nódulo sinoauricular, miocardio auricular, nódulo auriculoventricular o fascículo auriculoventricular. Algunos tipos son: fibrilación auricular, aleteo auricular, taquicardia sinusal, taquicardia auricular, taquicardia de reentrada del nódulo auriculoventricular y síndrome de Wolff-Parkinson-White.

   b) ***Taquiarritmias ventriculares:*** Tienen su origen, como su nombre lo dice, a nivel ventricular. Algunas de estas arritmias son: extrasístoles ventriculares, taquicardia ventricular, taquicardia ventricular idiopática, taquicardia ventricular en la displasia ventricular derecha arritmogénica, taquiarritmias ventriculares hereditarias, ritmo idioventricular acelerado, fibrilación ventricular y aleteo ventricular.

Por otro lado, los mecanismos a través de los cuales se generan las arritmias son [6]:

I. **Trastornos en la formación del impulso:**

   a) ***Automatismo normal:*** Algunas células cardíacas tienen la capacidad de generar automáticamente impulsos eléctricos, como las del nodo sinoauricular, nodo auriculoventricular, haz de His y fibras de Purkinje. La potenciación o inhibición de esta actividad puede resultar en arritmias clínicas.

   b) ***Automatismo anormal:*** Las células miocárdicas que en condiciones normales no poseen automatismo, generan impulsos eléctricos.

   c) ***Actividad desencadenada por pospotenciales tardíos:*** Al haberse completado una repolarización se produce una oscilación de voltaje de membrana, que a su vez puede generar un nuevo potencial de acción.

   d) ***Actividad desencadenada por pospotenciales precoces:*** Durante la fase de meseta de un potencial de acción o durante la fase de repolarización tardía se

genera una oscilación de voltaje de membrana, provocando cambios en la duración y perfil del potencial de acción

II. Trastornos en la conducción del impulso:

a) ***Retrasos y bloqueos:*** Funcionamiento defectuoso en la propagación del impulso eléctrico, por múltiples motivos, como la frecuencia cardíaca, tono del sistema nervioso autónomo, fármacos o procesos degenerativos.

b) ***Reentrada:*** Algún componente del sistema de conducción no se despolariza en conjunto con los otros y puede actuar como vínculo para volver a excitar zonas ya recuperadas de la despolarización. La reentrada puede ser anatómica o funcional.

La forma más ampliamente utilizada en el diagnóstico de arritmias es la técnica de electrocardiografía, que corresponde a la representación gráfica de la actividad eléctrica del corazón obtenida a partir de electrodos posicionados en la piel del paciente. Para realizar el registro se utilizan habitualmente dos electrodos por cada derivación, uno positivo y uno negativo, además de un electrodo adicional usado como referencia [7]. Una diferencia de potencial entre dos superficies de diferentes cargas forma un dipolo, cuya magnitud, polaridad y dirección se puede representar a través de un vector; el ECG registra los vectores instantáneos producidos por las diferencias de potencial entre los electrodos durante la actividad cardíaca, generando una gráfica positiva cuando la dirección del vector se acerca al electrodo positivo, una gráfica negativa cuando se aleja y ninguna cuando no se produce diferencia de potencial. Existen 12 configuraciones estándar, también llamadas derivaciones, para el posicionamiento de electrodos a utilizar: derivaciones de las extremidades (I, II y III), derivaciones de las extremidades aumentadas (aVR, aVL y aVF) y derivaciones precordiales (V1-V6) [8].

Según Palma et al. [7], existen 3 formas de registrar un ECG. Una de ellas es en tiempo real, donde un experto analiza presencialmente la señal en el instante en que el paceinte la genera. Otra forma de realizarlo es en la modalidad intermitente, que se registra la señal cuando se detecta algún evento particular. Finalmente, existe la forma continua de registrar un

ECG, donde se almacena el electrocardiograma (ECG) durante un periodo extenso de tiempo, llamándose también holter de arritmias cuando se realiza por 24 o 48 horas; éste es el método que más se utiliza en la práctica clínica.

Una vez que se dispone del ECG de un paciente, un experto (generalmente un cardiólogo especializado) debe analizar cada uno de los latidos y realizar un diagnóstico del paciente basándose en los patrones que caracterizan a los latidos normales o anormales. Dado que ésta corresponde a una tarea muy demandante de tiempo, para el análisis de la señal se utilizan programas computacionales que realizan un procesamiento previo y entregan información de interés. Sin embargo, el personal clínico debe revisar cuidadosamente los resultados y corregir errores que todos los sistemas de análisis poseen [7].

Un latido cardíaco se origina en el nodo sinoauricular (SA), el impulso generado en él viaja a través de la aurícula derecha simultáneamente hacia la aurícula izquierda y hacia el nodo auriculoventricular (AV). Luego el impulso se dirige al haz de His y se disemina hacia el miocardio ventricular a través de las fibras de Purkinje [9]. Esta actividad eléctrica fisiológica se ve reflejada en el ECG a través de diferentes puntos fiduciales, siendo las más relevantes: depolarización de las aurículas (onda P), depolarización de los ventrículos (complejo QRS) y repolarización de los ventrículos (onda T); a partir de las cuales se puede obtener medidas de relevancia clínica, como los intervalos PQ y QT. En la Figura 1.1 se muestran los puntos fiduciales y las características clínicas que se desprenden de ellos, mientras que en la Tabla 1.1 se despliegan los valores normales para algunas características.

Por otro lado, se proyecta que la fuerza de trabajo de los cardiólogos estará en crisis para mitad de siglo y que no alcanzará a cubrir las necesidades de los pacientes con ECV. Se estima que la demanda se aumentará por: el envejecimiento de la población, el crecimiento económico, expansión de la cobertura de seguros médicos y avances tecnológicos; mientras que la oferta de los mismos se verá disminuida por una posible disminución en la oferta de programas de entrenamiento cardiovascular y por un retiro precoz del mundo laboral [10].

En los Estados Unidos, en 2007, se presentó una prevalencia de 1.793 médicos por cada 100.000 habitantes, mientras que para los cardiólogos fue de 48 por cada 100.000

habitantes; se destaca además, la distribución no homogénea entre las zonas rurales y metropolitanas, mostrando una carencia y abundancia de cardiólogos, respectivamente [11].

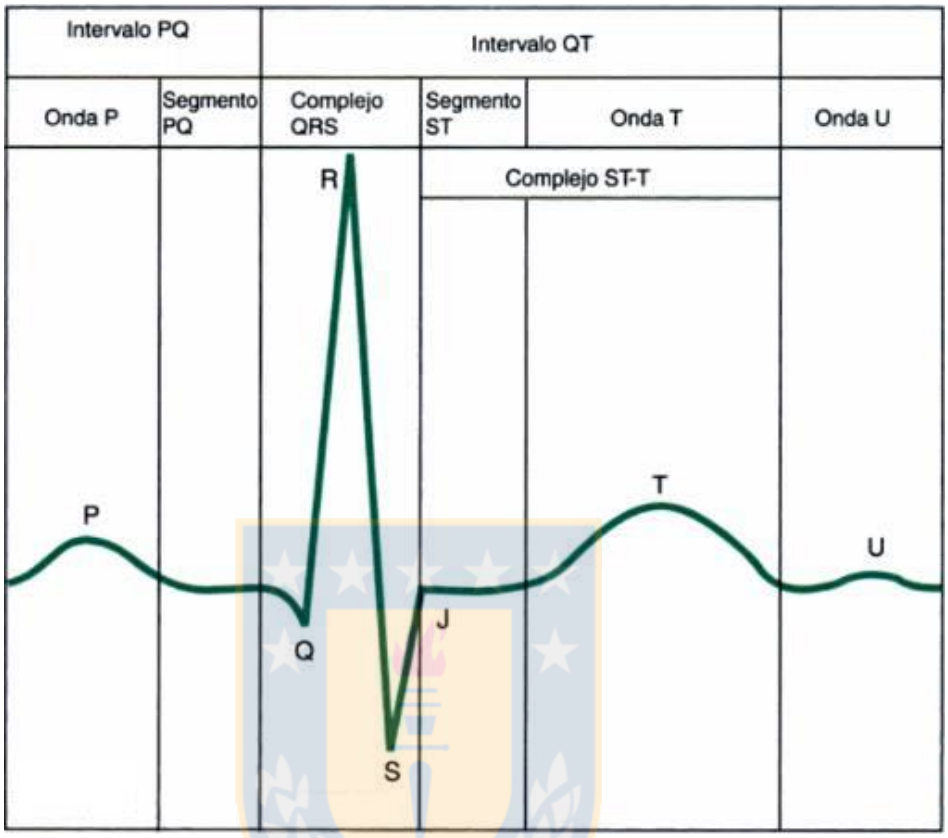

Figura 1.1: Puntos fiduciales (PQRSJTU) de un ECG normal y medidas de interés clínico en derivación II de las extremidades. Fuente: [5]

Característica	Valor normal	Límite normal
Duración de onda P	110 [ms]	±20 [ms]
Intervalo PQ/PR	160 [ms]	±40 [ms]
Duración del complejo QRS	100 [ms]	±20 [ms]
Intervalo QT	400 [ms]	±40 [ms]
Amplitud de onda P	0,15 [mV]	±0,05 [mV]
Altura de complejo QRS	1,5 [mV]	±0,5 [mV]
Nivel del segmento ST	0 [mV]	±0,1 [mV]
Amplitud de onda T	0,3 [mV]	±0,2 [mV]

Tabla 1.1: Valores normales de características del ECG en derivación II de las extremidades a 60 latidos por minuto en un hombre adulto. Fuente: [12]

Por su parte, en Chile, según datos del 2011, se dispone de 178,9 médicos por cada 100.000 habitantes, correspondiente a 29.996 profesionales. De estos, 15.668 tienen una especialidad médica y sólo 325 la tienen en cardiología (1,94 por cada 100.000 habitantes). Además la distribución geográfica de cardiólogos en Chile es: Norte 5,23%, Centro 81,23%, Centro-Sur 11,38% y Sur con 2,15%; a su vez, esto corresponde a 0,09; 0,26; 0,11 y 0,07 cardiólogos por cada 10.000 habitantes respectivamente [13].

Dado que los registros de ECG pueden contener cientos o miles de latidos, la revisión manual de un ECG aumenta la posibilidad de error humano por fatiga [14]. En respuesta a esta problemática se han enfocado esfuerzos en el desarrollo de diferentes sistemas automáticos de clasificación de latidos desde registros de ECG, que constan en la asignación de una etiqueta a cada latido (Ej: fibrilación auricular). Tanto en la práctica clínica como en el cuidado previo a la atención son necesarios algoritmos de análisis de ECG eficaces y rápidos, pues existe fuerte evidencia de mejoras sustanciales en la evolución de los pacientes al aplicar tratamiento de manera temprana [15]. El estado del arte de los sistemas automáticos de clasificación de latidos con técnicas de aprendizaje de máquina, se presenta en el siguiente capítulo.

El desarrollo de algoritmos efectivos de clasificación de latidos es un desafío mayúsculo, pues el ECG es un tipo de información de alta dimensión y que posee patrones complejos. Jambukia et al. presentaron algunos de los problemas a considerar al implementar clasificadores [16]:

I. **Falta de estandarización en las características del ECG**, para entrenar a los modelos, siendo que esto puede afectar seriamente el rendimiento de los algoritmos.

II. **Variabilidad de las características del ECG**, de acuerdo a condiciones fisiológicas o mentales, como por ejemplo, al variar la frecuencia cardiaca, puede cambiar la morfología de la señal de ECG.

III. **Individualidad de los patrones de ECG**, en cuanto a que éstos sean útiles y escalables para grandes bases de datos, considerando que los patrones encontrados en el conjunto de prueba presentan similaridad intraclase y variabilidad interclase.

IV. **No existen reglas óptimas** de clasificación para el ECG.

V. **Los pacientes pueden tener diferentes formas de onda de ECG**, como por ejemplo, diferir en amplitudes, tiempos, pendientes, etc. entre uno y otro paciente.

VI. **Variaciones entre los latidos de un mismo ECG**, así, un paciente puede presentar latidos normales y latidos arrítmicos de diferentes tipos en un mismo registro.

VII. **Encontrar el clasificador más apropiado**, pues el rendimiento del mismo depende de muchos factores como tipos de arritmias, diversidad de arritmias, base de datos utilizada para entrenar el modelo, métodos de extracción de características usados, etc.

Según De Chazal et al., las arritmias pueden ser divididas en dos grupos. El primero que incluye fibrilación ventricular y los tipos de taquicardia que colocan en riesgo la vida de los pacientes, siendo inmediatamente necesario el uso de un desfibrilador; la detección de arritmias de estos tipos han sido bien investigadas y son eficazmente detectadas. El segundo grupo consta de las arritmias que no son directamente un riesgo para la vida de los pacientes, pero que de no ser tratadas a tiempo pueden provocar problemas mayores; para la clasificación de arritmias de este grupo aún se están desarrollando algoritmos apropiados [17].

Velic et al. concluyen que la complejidad asociada al análisis de la señal ECG es la razón por la cual aún no se dispone de un algoritmo automático apropiado para otorgar soporte en la toma de decisiones del personal clínico [15].

En resumen, para finalizar esta introducción, se tiene que la creciente incidencia de enfermedades cardiovasculares sumada a una escasez de médicos especialistas, provocan que la revisión manual de registros electrocardiográficos (ECG) sea una tarea que no concuerda con el escenario actual. Este trabajo apunta hacia la automatización de procesos clínicos como la clasificación de latidos de la señal ECG de un paciente, que es un problema de gran complejidad y que aún presenta brechas por cubrir.

1.1. Hipótesis

Utilizando redes de aprendizaje profundo para la clasificación de latidos desde registros de ECG se pueden obtener resultados comparables a otros métodos utilizados en la literatura.

1.2. Objetivos

1.2.1 Objetivo general

Implementar un sistema de clasificación de latidos cardíacos desde ECG de adultos utilizando arquitecturas de redes de aprendizaje profundo.

1.2.2 Objetivos específicos

- Descargar la base de datos de arritmias del MIT-BIH completa.
- Implementar un algoritmo para el pre-procesamiento de las señales de ECG.
- Implementar un algoritmo para segmentar los registros de ECG en latidos individuales.
- Implementar una arquitectura de redes de aprendizaje profundo para obtener representaciones de los latidos de manera automática.
- Implementar clasificadores basados en redes neuronales que permitan la adopción del paradigma inter-paciente.
- Generar conjuntos de instancias para entrenar y probar las redes de aprendizaje profundo.
- Evaluar el rendimiento del modelo propuesto para compararlo con otros de la literatura.

1.3. Alcances y limitaciones

Los modelos de clasificación basados en aprendizaje profundo serán entrenados y evaluados utilizando la base de datos de arritmias del MIT-BIH y siguiendo las recomendaciones de la AAMI y de De Chazal et al. [17] para una comparación justa de los métodos en la literatura.

Capítulo 2. Revisión de literatura

2.1. Métodos tradicionales de clasificación de latidos cardíacos

Un sistema completamente automático de clasificación de latidos de ECG puede dividirse en 4 etapas secuenciales: preprocesamiento, segmentación de latidos, extracción de características, y clasificación. Se adopta esta misma forma para presentar la revisión de la literatura en esta sección.

2.1.1 Preprocesamiento de la señal

La señal de ECG se encuentra contaminada con ruidos de diversa índole, los que pueden ser clasificados como [12]:

I. Interferencia de la red eléctrica en 50 ± 2 Hz (o 60 Hz)
II. Ruido por pérdida de contacto entre la piel y los electrodos
III. Perturbaciones por el movimiento del paciente y los electrodos
IV. Ruido electromiográfico por la actividad eléctrica de los músculos
V. Desvío de línea base, usualmente provocado por la respiración
VI. Ruido del dispositivo que realizó el registro de ECG
VII. Ruido generado por otros equipos eléctricos
VIII. Ruido de cuantización y muestreo
IX. Perturbaciones por el procesamiento de la señal

El personal clínico tiene la capacidad de lidiar con señales que presentan ruido, pero los sistemas automáticos de clasificación son negativamente afectados, por lo que es necesario atenuarlos o eliminarlos mediante el filtraje o eliminación de señales [18]. Así, se han utilizado diferentes métodos, como filtros de fase lineales pasa-bajos y pasa-altos para manejar los ruidos, mientras que para el ajuste de la línea base se han ocupado filtros de mediana, filtros de fase lineales pasa-altos, filtros de mediana promedio, etc. [16].

En la revisión bibliográfica de Luz et al. [14] se encontró que los filtros digitales recursivos de tipo respuesta de impulso finito (FIR) funcionan bien para atenuar frecuencias conocidas, como la frecuencia de la red eléctrica, pero aplicar varios filtros sin conocer las

frecuencias a atenuar puede hacer que la señal sea inútil para el diagnóstico de enfermedades cardíacas. Para este último caso, se han implementado filtros adaptativos de redes neuronales obteniendo buenos resultados [14].

Los métodos basados en la transformada de *Wavelet* entregan buenos resultados y son fáciles de implementar. Una variación de éstos, es la transformada *wavelet biónica multi adaptativa* que ha entregado aún mejores resultados [14].

Otros métodos útiles: filtros bayesianos no lineales y filtros de Kalman extendidos (son los que presentan mejor efectividad hasta el momento). Los resultados comúnmente se expresan en razón señal-ruido [14].

Luz et al. [14] explican que existe una carencia de trabajos con evaluación del impacto del preprocesamiento de la señal de ECG en el resultado de los clasificadores posteriormente usados, por lo que proponen este tema como un área de investigación.

2.1.2 Segmentación de los latidos

Antes de extraer características desde la señal de ECG, la serie de tiempo debe ser segmentada en latidos individuales utilizando como referencia el complejo QRS. Existen muchos métodos para detectar los puntos fiduciales del ECG, como filtros digitales o la transformada de *Wavelet*; incluso existen herramientas como "ecgpuwave" útiles en la detección de los mismos [19].

Esta etapa ha sido estudiada desde más de 30 años y muchos algoritmos han sido desarrollados para este fin. Un enfoque ampliamente usado es la utilización de: filtros digitales para atenuar los ruidos y la línea base, traslaciones no lineales para realzar el peak R y umbrales de detección adaptativos, como el algoritmo de Pan Tompkins [15]. Existen métodos más sofisticados como redes neuronales, algoritmos genéticos, transformada de wavelet, bancos de filtros, Quad level vector y otros. Algunos algoritmos identifican las ondas P y T del ECG, agregando más información sobre los latidos cardíacos [14].

La segmentación es una etapa importante, pues los errores en ésta se propagan hacia las siguientes etapas, pero no está claro de qué manera, ya que la evaluación del impacto de los diferentes métodos de segmentación en la clasificación de arritmias es un campo de investigación poco explorado [14].

2.1.3 Extracción de características

Una característica corresponde a cualquier información extraída de un latido cardíaco usada para determinar cuál es el tipo de latido. Esta etapa se puede subdividir en la extracción misma de características y en la selección de las más representativas.

I. Extracción de características

Las características pueden ser extraídas desde la morfología de la señal en el dominio del tiempo y/o de la frecuencia, o desde el ritmo cardíaco [14]. Según Faust et al., los métodos de extracción de características pueden ser clasificados en lineales y no lineales [18]:

a) ***Métodos lineales:*** Intervalo RR, espectro en frecuencia, gráfico de Lorenz, estadísticos de primer y segundo orden, transformada discreta de Wavelet (DWT), análisis de componentes independientes (ICA), análisis de componentes principales (PCA), análisis de discriminante lineal (LDA) y la transformada discreta del coseno (DCT).

b) ***Métodos no lineales:*** Características fractales, agrupamiento por expectación-maximización y estadísticos de alto orden.

Jambukia et al. agregan a los anteriores, los siguientes métodos: Transformada continua de Wavelet (CWT), transformada S, transformada discreta de Fourier (DFT), wavelet daubechies y el algoritmo de Pan Tomkins [16].

La característica más comúnmente utilizada es el intervalo RR. Variaciones de éste son usadas para reducir el ruido, como el promedio de RR en un intervalo de tiempo, que ha demostrado entregar mejores resultados en la etapa de clasificación. Otra característica

que ha sido usada es la duración del complejo QRS. Uno de los métodos que entrega mejor exactitud es la extracción de características de los dominios del tiempo y frecuencia en conjunto con las extraídas del intervalo RR [14].

Para reducir la dimensión del vector de características (considerándose como el conjunto de muestras de un latido) se han empleado diferentes técnicas en la vecindad del *peak* R, como análisis de componentes principales (PCA) y análisis de componentes independientes (ICA). PCA es mejor para reducir el ruido e ICA es mejor para la extracción de características, y su uso combinado presenta mejores resultados que su uso individual. El análisis de componentes principales con kernel (KPCA) tiene un rendimiento superior a PCA por su estructura no lineal. Otras técnicas utilizadas para la reducción de características son agrupamiento (*clustering*), análisis de discriminante generalizado (GDA), interpolación, proyecciones aleatorias, codificación predictiva lineal, acumulados de alto orden, correlación de dimensión y mayor exponente Lyapunov, transformada Hermite y dimensión de fractal local [14].

La mayoría de los métodos anteriormente mencionados utilizan la transformada de *Wavelet*, ya que es considerada como la mejor opción para extraer información desde los dominios del tiempo y frecuencia. Los parámetros relevantes de esta transformada son la función wavelet madre, el orden del filtro y el nivel de descomposición; y se pueden optimizar usando optimización por enjambre de partículas (PSO). Para la clasificación de latidos cardiacos se ha logrado el mejor resultado en exactitud utilizando la función wavelet madre Daubechies de orden 2 [14].

Zhang et al. extrajeron características en el dominio del tiempo y las clasificaron en: intervalos inter-latidos, intervalos intra-latidos, amplitudes morfológicas, áreas morfológicas y distancias morfológicas; todas basadas en la identificación de puntos fiduciales del ECG [19].

I. Selección de características

La importancia de esta etapa radica en que los algoritmos de clasificación más efectivos sólo pueden lidiar con una cantidad limitada de parámetros y múltiples trabajos

en el campo del aprendizaje de máquina sugieren mantener la mínima cantidad posible de estos [18].

Pocos autores han investigado sobre métodos para seleccionar características en la clasificación de arritmias. La selección de características puede tener beneficios en la clasificación de arritmias, como aumentar el poder de generalización de los algoritmos de clasificación y como la reducción del costo computacional [14].

Uno de los métodos utilizados para este fin es la búsqueda secuencial flotante que muestra mejores resultados que los de la literatura y utilizando sólo 8 características. Otros métodos que han demostrado entregar buenos resultados son los algoritmos genéticos (GA) y la optimización por enjambre de partículas (PSO). En uno de los trabajos incluidos en la revisión bibliográfica de Luz et al. [14] identificaron como características más importantes los intervalos RR, la amplitud y duración de la onda T, y estadísticos de segundo orden. Además, los autores declaran que sería de gran importancia comprender de mejor forma la correlación entre enfermedades cardíacas y características extraídas del ECG.

2.1.4 Clasificación

La finalidad de los algoritmos automáticos de clasificación es aprender los patrones de cada clase (tipo de latido) basado en un vector de características y luego utilizar este conocimiento cuando se presenta un nuevo elemento (nuevo latido) que no posee clase asignada. El proceso de utilizar datos con etiquetas conocidas en el entrenamiento de los modelos, se denomina aprendizaje supervisado; y en caso contrario, aprendizaje no supervisado [18]. Algunos de los métodos utilizados en la clasificación de latidos mediante aprendizaje de máquina son:

I. Máquinas de soporte vectorial (SVM)

Convierte los patrones del vector de entrada a un espacio de características con mayores dimensiones, a través de mapeos no lineales, y obtiene un hiperplano de

separación óptimo entre las clases. Para el mapeo no lineal se utilizan transformaciones de kernel como las funciones cuadrática, polinomial y de base radial (RBF) [18], [20].

Según Luz et al. [14], este algoritmo de aprendizaje supervisado es uno de los más utilizados en la clasificación de latidos cardíacos. En su revisión bibliográfica se consideran diferentes variaciones de las SVM: combinadas con lógica difusa, combinadas con un conjunto de clasificadores, algoritmos genéticos con SVM difusas restringidas y SVM con mínimos cuadrados. Dado que SVM presenta un mal comportamiento frente a desbalances de clases, se han implementado algunas formas de manejarlo, como utilizar SVM con estructura jerárquica o SVM ponderado por cada clase, pero Luz et al. advierten que es un problema poco estudiado y lo proponen como un tema de investigación futura.

II. Redes neuronales artificiales (ANN)

Cada neurona calcula la suma ponderada de sus entradas e introduce esta suma en una función no lineal llamada función de activación. En general se utilizan redes neuronales con alimentación hacia adelante y su rendimiento depende del número de capas ocultas, número de neuronas ocultas, algoritmo de aprendizaje de la red y función de activación de cada neurona. Para el entrenamiento se puede ocupar el método de propagación hacia atrás, que calcula el error cuadrático medio entre la respuesta obtenida y la deseada, y ajusta los pesos de la red hasta cumplir un criterio de término [20].

Las ventajas de las ANN son: se auto enseñan o auto adaptan, pueden modelar problemas no lineales, son rápidas de entrenar, muestran una alta exactitud, son robustas a ruido y son fácilmente escalables. Mientras que sus desventajas son que los algoritmos de entrenamiento no aseguran que se alcanza el óptimo global y que no necesariamente puede encontrar un óptimo en problemas de muy altas dimensiones [16].

Las arquitecturas de ANN más usadas en la clasificación de latidos son los perceptrones multicapa (MLP) y las redes neuronales probabilísticas (PNN), siendo las segundas las más robustas y computacionalmente eficientes; sin embargo, se han propuesto numerosas configuraciones de ANN en la literatura [14].

III. Discriminantes lineales

Son métodos estadísticos basados en funciones discriminantes, las cuales son estimadas a partir de un conjunto de entrenamiento al que se trata de separar linealmente al ajustar un vector de pesos y un sesgo. Las ventajas de los discriminantes lineales radican en que pueden manejar los desbalances de clases y que requieren un menor tiempo computacional para su entrenamiento (al no ser un método iterativo). Sin embargo, MLP ha demostrado un rendimiento notoriamente superior a este método [14].

IV. Computación de reserva

Corresponden a modelos dinámicos que buscan procesar una serie de tiempo, representando la señal a través de una reserva dinámica no adaptable y aplicando una lectura dinámica a dicha reserva. Esta técnica es robusta al desbalance de clases, posee un bajo costo computacional y obtiene los mejores resultados en los trabajos revisados por Luz et al [14].

V. Otros

El uso del algoritmo de los k vecinos cercanos (kNN) conlleva a un alto costo computacional en la etapa de prueba, por lo que su utilidad es limitada en escenarios de la vida real. Por su parte, las técnicas de agrupamiento se han utilizado en combinación con ANN con el fin de mejorar la capacidad de generalización y tiempo de aprendizaje de estas últimas. Se ha utilizado además: modelos ocultos de Markov, clasificadores de hipercubo, bosques de ruta óptima y campos condicionales aleatorios. Los árboles de decisión no han sido muy utilizados dado que sólo pueden manejar un número reducido de características y los métodos basados en reglas son los que presentan peor rendimiento [14].

Faust et al. exponen otros clasificadores con aprendizaje supervisado, como: ADABoost (sistema robusto de decisiones a partir de varios sistemas débiles de aprendizaje), clasificadores difusos, modelos mixtos gausianos, árboles de regresión y clasificación, bosques aleatorios y clasificación binaria por umbrales [18].

2.2. Clasificación de latidos con aprendizaje profundo

A continuación se presentan trabajos enfocados en la clasificación de latidos cardíacos desde señales de ECG utilizando técnicas de aprendizaje profundo.

Yan et al. [21] implementaron una red de creencia profunda (DBN) basada en máquinas de Boltzmann restringidas (RBM) para realizar clasificación de latidos desde un ECG. Se utilizó la BD de arritmias del MIT-BIH considerando las 2 derivaciones de cada registro y realizaron la clasificación para 12 clases de latidos, sólo preprocesando el desvío de la línea base. Las RBM se utilizaron para generar las representaciones de las señales de entrada y fueron entrenadas con el método del gradiente estocástico descendente. Para formar la DBN de 5 capas se apilaron varias RBM y finalmente se agregó un clasificador softmax; para el ajuste fino se utilizó el algoritmo de optimización de cuasi-Newton llamado L-BFGS. Los resultados obtenidos son Acc=98,83%, Se=99,83 y Sp=96,05%; aunque cabe señalar que no se especifica sobre las señales que se utilizaron para entrenar y evaluar el modelo.

Huanhuan et al. [22] realizaron la clasificación de latidos cardíacos utilizando DBN, redes neuronales (NN) y máquinas de soporte vectorial (SVM), dividiendo su implementación en 2 etapas: extracción de características y clasificación. Luego de pre-procesar los datos con Daubechie 8 de Wavelet, se conformó un vector de características a través de 2 formas: extrayendo características morfológicas de la señal (asociadas al intervalo RR) y representaciones a través de DBN de 4 capas basadas en RBM. Para la etapa de clasificación se utilizaron NN y SVM no lineales de manera separada para contrastar resultados. La NN posee 53 nodos en la capa de entrada, 200 neuronas en la capa oculta y 6 neuronas en la capa de salida (correspondiente a 6 clases de latidos), y fue entrenada con el método de propagación hacia atrás (BP). La SVM no lineal fue probada con 3 kernel diferentes: gausiano, polinomial y cuadrático. Se utilizó la BD del MIT-BIH considerando sólo la derivada modificada II de las extremidades y seleccionando al azar los latidos de entrenamiento y prueba. Los mejores resultados son obtenidos con el clasificador SVM no lineal con kernel gausiano que mostró un Acc=98,49%.

Kiranyaz et al. [23] presentan un sistema de clasificación de latidos basado en el

paradigma paciente-específico y utilizando redes neuronales convolucionales (CNN). En cuanto al pre-procesamiento, se submuestreó cada latido a 64 y 128 muestras y se generó otra representación de ellos usando la transformada rápida de Fourier (FFT). Para la extracción de características y clasificación, se utilizó una CNN de 1D con 3 capas unida a un perceptrón multicapa (MLP), que fueron entrenadas con el método de BP. Se utilizó la BD del MIT-BIH y se dividió en 20 pacientes para entrenamiento y 24 para prueba. Se seleccionó aleatoriamente (desde el conjunto de entrenamiento) un grupo de latidos para conformar un conjunto de entrenamiento común para todos los pacientes y se etiquetaron los latidos de los primeros 5 minutos de cada registro del conjunto de pruebas (específico para cada paciente). Los resultados obtenidos para las clases VEB y SVEB son respectivamente Acc=98,6%, Se=95%, P^+=89,5%, Acc=96,4%, Se=64,6% y P^+=62,1%.

Rahhal et al. [24] presentan un sistema de clasificación de latidos basado en el paradigma paciente-específico utilizando una arquitectura híbrida de aprendizaje profundo. La señal de ECG fue pre-procesada corrigiendo la línea base y quitando el ruido de la red eléctrica y de altas frecuencias, para más tarde submuestrearla y segmentarla en 50 muestras por latido. Para la extracción de características, se apilaron auto codificadores con eliminación de ruido (SDAE) y con restricciones de dispersión, para obtener así una representación de características de manera no supervisada; además, se incluyeron características morfológicas basadas en el intervalo RR. Luego, para la clasificación se utilizó una capa softmax, formando una red neuronal profunda (DNN). El modelo propuesto se ajusta a los patrones de cada paciente mediante la intervención de un humano experto que etiqueta los latidos más relevantes e inciertos, seleccionados según 2 criterios: entropía y desempates. Se utilizó la BD del MIT-BIH según las recomendaciones de la AAMI y además se utilizaron otras 2: BD INCART y BD de arritmia supraventricular del MIT. Los mejores resultados obtenidos para SVEB y VEB respectivamente son Acc=99,8%, Se=96,7%, P^+=99,3%, Acc=99,9%, Se=98,8% y P^+=99,7%.

Übeyli [25] utilizó redes neuronales recurrentes (RNN) para la clasificación de latidos desde la BD de Physionet, considerando 4 clases: latido normal, latido de falla cardíaca congestiva, latido de taquiarritmia ventricular y latido de fibrilación auricular. Para la extracción de características se utilizaron 3 métodos de eigenvectores (estiman frecuencias y

potencias de la señal): Pisarenko, clasificación de señal múltiple y norma-mínima. En cuanto a la clasificación se implementó una RNN que fue entrenada con el algoritmo de Levenberg-Marquardt y un perceptrón multicapa (MLP). Los resultados mostraron que la RNN es superior al MLP encuanto a sensibilidad y especificidad, mientras que en exactitud los resultados fueron respectivamente Acc=98,06% y Acc=90,83%.

2.3. Consideraciones en la clasificación de latidos

La asociación por el progreso de la instrumentación médica (AAMI) desarrolló una estandarización para la evaluación de los métodos de clasificación de latidos cardíacos, denominada ANSI/AAMI EC57:1998/(R)2008, donde se definen los protocolos para que los experimentos sean reproducibles y comparables con otros [26].

La mayoría de las bases de datos consideradas por la AAMI poseen etiquetas relacionadas al tipo de latido o a los eventos cardíacos, así como algunos puntos fiduciales del ECG. El estándar también hace referencia a la forma (o formato) en que se debe realizar dicho etiquetado [14]. A pesar de que existen muchos tipos de arritmias, la AAMI recomienda que sólo se detecten 15 clases, agrupadas en 5 súper clases, como se muestra en la Tabla 2.1. Cabe señalar que para el análisis de los métodos se recomienda no considerar los pacientes con marcapasos ni los segmentos que contienen *aleteo* o fibrilación ventricular.

El estándar de la AAMI recomienda el uso de las siguientes bases de datos (BD) [14]:

I. **BD de arritmias del Instituto de tecnología de Massachusetts- Hospital Beth Israel (MIT-BIH)** [27], [28]**:** Es la base de datos más utilizada en la literatura. Contiene 48 registros de ECG a 360Hz por aproximadamente 30 min de 47 pacientes adultos diferentes y se dispone de las etiquetas de los latidos de acuerdo al estándar de la AAMI. Cada uno posee 2 derivaciones de ECG, la derivación A (modificación de la deriv. II de las extremidades) y la derivación B (deriv. que puede cambiar entre V1, V2, V5 o V4).

II. **BD de ST-T de la Sociedad Europea de Cardiología (EDB)** [27], [29]**:** Contiene 90 registros de ECG a 250Hz y resolución de 12 bits, correspondientes a 79 pacientes que

padecen una enfermedad cardíaca específica (pues se creó la BD para el análisis del segmento ST y la onda T). Cada registro tiene una duración de alrededor de 2 horas y se presenta en 2 derivaciones. Además, posee las etiquetas de los latidos de acuerdo a los criterios de la AAMI.

III. **BD para la evaluación de detectores de arritmias ventriculares de la Asociación Americana del Corazón (AHA)**[30], [31]**:** Contiene 154 registros de ECG a 250Hz y resolución de 12 bits, mostrando 2 derivaciones. Cada registro tiene una duración de 3 horas, pero sólo los últimos 30 minutos poseen las anotaciones del tipo de latido y no cumplen con el estándar de la AAMI, dado que no se hace distinción entre ritmo sinusal normal y SVEB.

IV. **BD de taquiarritmia ventricular de la Universidad de Creighton** [27], [32]**:** Contiene 35 registros de ECG a 250Hz y 12 bits, con una duración de 8 minutos cada uno. Posee anotaciones, pero no utiliza todas las clases del estándar.

V. **BD de prueba de estrés ante el ruido** [27], [33]**:** Posee 3 y 12 registros de media hora con y sin ruido, respectivamente. Esta BD fue construida a partir de 2 señales de la BD del MIT-BIH, añadiéndoles interferencias comunes en la atención ambulatoria.

Las BD mencionadas poseen un claro desbalance de clases, por lo que se requieren varias medidas de desempeño para evaluar el rendimiento de los métodos de clasificación de latidos. Las recomendadas por la AAMI son: *recall* o sensibilidad (Se), predictividad positiva o precisión (P^+), tasa de falsos positivos (FPR) y exactitud global (Acc) [14], [26].

El estándar de la AAMI no contempla recomendaciones sobre cuáles latidos o pacientes deben ser considerados para entrenar el modelo de clasificación y cuáles deben ser usados en la fase de prueba. Se ha demostrado que utilizar los latidos de un mismo paciente en ambas fases genera sesgo en la evaluación del modelo, dado que los modelos aprenden particularidades del paciente en el entrenamiento y luego, en la fase de prueba, despliegan resultados muy optimistas [14], [17].

Grupo	Símbolo	Clase
N Cualquier latido no categorizado como SVEB, VEB, F o Q	N	Lat. Normal
	L	Lat. con bloqueo de la rama izquierda
	R	Lat. con bloqueo de la rama derecha
	e	Lat. de escape auricular
	j	Lat. de escape nodal
SVEB Lat. ectópico supraventricular	A	Lat. auricular prematuro
	a	Lat. auricular prematuro aberrado
	J	Lat. nodal prematuro
	S	Lat. supraventricular prematuro
VEB Lat. ectópico ventricular	V	Contracción ventricular prematura
	E	Lat. de escape ventricular
F Latidos de fusión	F	Fusión de latido ventricular y normal
Q Lat. desconocidos	P	Lat. de marcapasos
	f	Fusión de latido de marcapasos y normal
	U	Lat. inclasificable

Tabla 2.1: Principales tipos de latidos presentes en la base de datos de MIT-BIH. Fuente: [14]

Existen 3 enfoques para el desarrollo de los modelos de clasificación [14]:

I. **Intra-paciente:** Se utilizan latidos de un mismo paciente en las fases de entrenamiento y prueba, y se eligen aleatoriamente.

II. **Inter-paciente:** Los latidos son cuidadosamente divididos en conjuntos de entrenamiento y prueba, para evitar que los latidos de un paciente se presenten en ambos grupos.

III. **Paciente-específico:** Corresponde a un enfoque semi-automático de clasificación, en el que interviene un humano experto, como por ejemplo, generar grupos (clustering) con técnicas de aprendizaje no supervisado y luego ser etiquetados por expertos.

Los investigadores advierten que la mayoría de trabajos no se preocupa de utilizar el paradigma inter-paciente, provocando que la comparación entre métodos sea muy difícil. En esta línea proponen un esquema de trabajo para complementar el estándar de la AAMI y hacer los métodos fácilmente comparables [14]:

I. **Selección de BD:** Usar la de MIT-BIH bajo el esquema inter-paciente y para probar la capacidad de generalización utilizar la BD INCART.

II. **Pre-procesamiento:** Además de los métodos a usar, también es necesario probar el modelo con la señal sin filtrar y probarlo también con el pre-procesamiento realizado por De Chazal et al [17].

III. **Segmentación:** Agregar pequeñas distorsiones al reconocimiento de la onda R, para probar cómo se comporta el modelo.

IV. **Extracción de características:** Usar selección de características para evaluar cuáles son las que más aportan y utilizar selección de características diferenciada por cada clase.

V. **Clasificación:** Utilizar validación cruzada para el entrenamiento del modelo e investigar el impacto del desbalance de clases en el clasificador escogido, mostrando los resultados con y sin técnicas de compensación de desbalances.

VI. **Evaluación:** Presentar los resultados como lo sugiere el estándar de la AAMI.

Luz et al. concluyen que los enfoques semi-automáticos pueden mejorar los resultados en un 40% aún con pocos latidos utilizados, sin embargo, estos métodos requieren de la intervención de expertos. Además, agregan que un limitante importante en el avance de la clasificación de latidos se debe a la escasa cantidad de BD útiles para este fin [14].

Para terminar esta sección, en la Tabla 2.2, se presenta un resumen con algunos de los trabajos realizados bajo el paradigma inter-paciente que han demostrado los mejores

resultados en la literatura en cuanto a exactitud (Acc) y que corresponde a un extracto de la revisión bibliográfica de Luz et al. [14].

Trabajo	**Conjunto de características**	**Clasificador**	**Resultados**
Escalona-Moran et al.[34]	Señal en crudo	Computación de reserva	Acc= 98% Se_N= 96%; $+P_N$= 91% Se_S= 79%; $+P_S$= 96% Se_V= 96%; $+P_V$= 99%
Lin y Yang [35]	Intervalo R-R normalizado	Discriminante lineal ponderado	Acc= 93% Se_N= 91%; $+P_N$= 99% Se_S= 81%; $+P_S$= 31% Se_V= 86%; $+P_V$= 73%
Llamedo y Martinez [36]	*Wavelet*, vectorcardiograma + SFFS	Discriminante lineal ponderado	Acc= 93% Se_N= 95%; $+P_N$= 98% Se_S= 77%; $+P_S$= 39% Se_V= 81%; $+P_V$= 87%
Bazi et al. [37]	Morfológicas, *Wavelet*	SVM, IWKLR, DTSVM	Acc= 92%

Tabla 2.2 Mejores resultados de la literatura bajo el esquema inter-paciente. SFFS: Búsqueda flotante secuencial hacia adelante, SVM: Máquina de soporte vectorial, IWKLR: Regresión logística con kernel de importancia ponderada, DTSVM: SVM con transferencia de dominio. Fuente: Extracto de [14]

Capítulo 3. Marco teórico

En este capítulo se darán a conocer conceptos de interés para comprender de mejor forma la metodología utilizada, además de brindar un trasfondo teórico de la misma.

3.1. Aprendizaje de máquina (Clasificación)

El aprendizaje de máquina es una rama de estudio de las ciencias de la computación que se enfoca en modelos computacionales que pueden aprender y hacer predicciones de un conjunto de datos. El aprendizaje de máquina puede ser dividido en 2 grandes grupos: aprendizaje no supervisado y supervisado. El primero tiene como objetivo describir la estructura oculta o relaciones entre los datos de entrada sin etiquetar [38]. El segundo grupo se refiere a los modelos que infieren una función a partir de datos de entrenamiento (o ejemplos) etiquetados, es decir, datos que cuentan con un arreglo de entrada y un valor de salida deseado. Cabe señalar que algunos autores añaden otros 2 grupos: Aprendizaje semisupervisado, donde se entregan datos de entrenamiento incompletos al modelo y Aprendizaje por refuerzo, en el cual el algoritmo persigue un objetivo y recibe como retroalimentación recompensas y castigos.

Los algoritmos supervisados se pueden dividir a su vez en regresión o clasificación. La primera tiene como salida valores contínuos, mientras que la otra tiene clases o valores categóricos. El objetivo de la clasificación es aprender los patrones de diferentes clases (número finito de categorías) basándose en un vector de características conocido, para luego aplicar este conocimiento adquirido cuando aparezca un nuevo vector de características desconocido y asignarle una clase [18]. Durante la fase de aprendizaje, los modelos de clasificación ajustan los parámetros según un indicador de desempeño o función objetivo, por lo que un problema de clasificación puede ser visto también como un problema de optimización [39].

Para una evaluación justa de los modelos de clasificación se utilizan diferentes conjuntos de datos para las etapas de entrenamiento y prueba, pues lo que se busca es conocer la habilidad de generalización de los mismos, es decir, la capacidad de encontrar patrones en

los datos de entrenamiento que sirvan para asignar correctamente una clase a casos nuevos, que no han sido explicitamente entregados al modelo.

Cuando un clasificador se ajusta mucho a los datos de entrenamiento pierde habilidad de generalización y se produce el fenómeno de sobre-ajuste, que hace referencia a un modelo que describe ruido o error aleatorio en vez de las relaciones subyacentes de los datos de entrenamiento, provocando que tenga un excelente desempeño con los datos de entrenamiento, pero un bajo rendimiento de predicción para datos nuevos. En la Figura 3.1 se muestra un modelo polinomial de diferentes grados para mostrar gráficamente el efecto del sobre-ajuste; el de grado 1 presenta sub-ajuste, pues no es capaz de representar los datos, el de grado 4 se aproxima a la función correcta y el de grado 15 presenta sobre-ajuste, pues representa características particulares de los datos. Además se muestra el error cuadrático medio (MSE) en la predicción del conjunto de prueba, evidenciando que el modelo sobre-ajustado posee el mayor error y por consiguiente el peor rendimiento [40].

Los resultados de la implementación de modelos de clasificación pueden ser expresados a través de una matriz de confusión, que corresponde a una matriz cuadrada en la que se contrastan las clases predichas por el modelo de clasificación con las clases que realmente corresponden a los elementos. En la Tabla 3.1 se muestra una matriz de confusión para un problema de clasificación binario.

Considerando respectivamente TP y TN verdaderos positivos y negativos, y FP y FN falsos positivos y negativos. De la tabla anterior se desprenden medidas de desempeño del estándar, como: Sensibilidad, Predictividad Positiva, Tasa de Falsos Positivos y Exactitud; que se calculan respectivamente según las ecuaciones (3.1)-(3.4) [14].

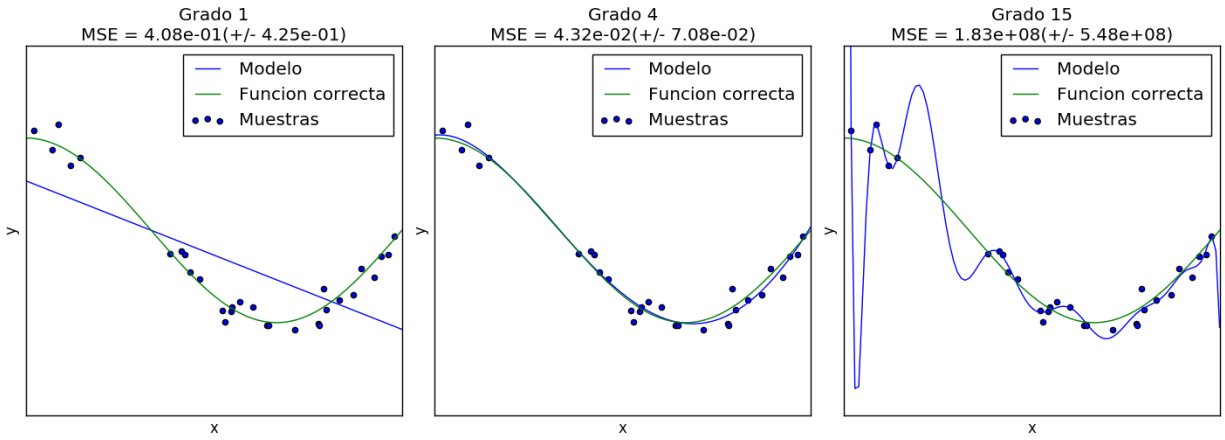

Figura 3.1 Modelo polinomial con sub-ajuste, con el ajuste correcto y con sobreajuste, respectivamente. Fuente: [40]

		Clasificación	
		Positivo	Negativo
Real	Positivo	TP	FN
	Negativo	FP	TN

Tabla 3.1 Matriz de confusion para 2 clases (Positivo y negativo). Fuente: Elaboración propia

$$Se = \frac{TP}{TP+FN} \tag{3.1}$$

$$P^{+} = \frac{TP}{TP+FP} \tag{3.2}$$

$$FPR = \frac{FP}{FP+TN} \tag{3.3}$$

$$Acc = \frac{TP+TN}{TP+FP+FN+TN} \tag{3.4}$$

3.2. Aprendizaje profundo

El aprendizaje profundo o *Deep learning* (DL), corresponde a técnicas de aprendizaje de máquina en las que muchas capas de unidades procesadoras de información son utilizadas en el aprendizaje no supervisado de características y para el análisis o clasificación de patrones (aprendizaje supervisado) [41]. La esencia de DL es obtener múltiples niveles de representación, a partir de módulos simples no lineales que transforman la representación de un nivel a otro más alto y abstracto. Para efectos de clasificación, las capas de representación más altas amplifican los aspectos de la entrada que son más importantes para discriminar entre las clases y suprimen las variaciones que son irrelevantes [42].

La profundidad de la arquitectura hace referencia al número de niveles de operaciones no lineales aprendidos. Mientras los algoritmos comúnmente utilizados en aprendizaje de máquina corresponden a arquitecturas superficiales (de 1 a 3 niveles), el cerebro de los mamíferos está organizado en una arquitectura de tipo profunda, ya que para un estímulo de entrada se generan múltiples niveles de abstracción. Por esta razón los investigadores de redes neuronales artificiales han dedicado esfuerzos en replicar este tipo de arquitecturas [43].

Las redes neuronales profundas han sido exitosamente aplicadas en clasificación, regresión, reducción de dimensiones, modelamiento de texturas, modelamiento de movimiento, segmentación de objetos, recuperación de información, robótica, procesamiento de lenguaje natural, filtrado colaborativo, reconocimiento de imágenes, reconocimiento de voz, procesamiento de señales y otros [42]–[44].

Las arquitecturas profundas se pueden clasificar en [41], [44]:

I. **Arquitecturas profundas generativas:** Son usadas para caracterizar las propiedades de correlación de alto orden desde la información observada, con la finalidad de realizar análisis o síntesis de patrones y/o para caracterizar las distribuciones estadísticas entre la información observada y las clases asociadas; de esta manera, se asocia este tipo de arquitectura con aprendizaje no supervisado. Algunos subtipos son: los modelos profundos basados en la energía (incluyendo autocodificadores), máquinas

de Boltzmann profundas (DBM), redes de creencia profunda (DBN), redes de suma-producto (SPN) y algunos tipos de redes neuronales recurrentes (RNN).

II. **Arquitecturas profundas discriminativas:** Son utilizadas para obtener poder discriminativo en la clasificación de patrones y se asocian a aprendizaje supervisado. Algunos tipos son: redes neuronales profundas (DNN), redes apiladas profundas (DSN), algunos tipos de RNN y redes neuronales convolucionales (CNN).

III. **Arquitecturas profundas híbridas:** Tienen la misma meta que las discriminativas, pero siendo asistidas por las generativas. Éstas pueden ayudar en la inicialización de las discriminativas (optimización) en forma de pre-entrenamiento y/o en el control de la complejidad (regularización) del modelo completo. Algunos modelos de este tipo son: DNN pre entrenadas con DBN y CNN pre entrenadas con DBN.

La elección de la arquitectura de DL es trascendental para el éxito o fracaso de la aplicación en la que se desea utilizar, sin embargo las ventajas y desventajas de cada arquitectura no se entienden a cabalidad y no existe una metodología para encontrar la que mejor se ajuste al problema, por lo que para obtener buenos resultados es importante estar consciente de sus capacidades, de las características de los datos a usar y de los objetivos de la investigación. Las capacidades de una arquitectura hacen referencia a las aplicaciones en las que fueron útiles, por ejemplo: DNN en el análisis de correlaciones en datos de grandes dimensiones, CNN en análisis de información espacial y RNN en análisis de información secuencial.

Una vez elegida la arquitectura, existe una gran cantidad de hiperparámetros que configurar y que afectan fuertemente los resultados, algunos de estos son: el número de capas, el número de unidades escondidas, los valores de los pesos en la iniciación, iteraciones de aprendizaje, tasa de aprendizaje y otros; por mucho tiempo éstos han sido ajustados por un experto, pero últimamente han surgido algoritmos para optimizar su ajuste [45].

3.3. Redes neuronales artificiales

Las redes neuronales artificiales fueron creadas en un intento de imitar el funcionamiento de las redes neuronales biológicas, las cuales presentan unidades de procesamiento (neuronas) interconectadas (sinapsis), que procesan información de múltiples entradas (dendritas) para obtener una única salida (axón), en base a la fuerza sinaptica de cada conexión axón-dendrita y a la susceptibilidad de disparo de la neurona.

En el modelo artificial de una neurona, las salidas de las anteriores (X) interactúan con las dendritas a través de una ponderación o peso (W) que es aprendido por el modelo, de manera de controlar la influencia de X tanto excitatoria (pesos positivos) como inhibitoriamente (pesos negativos). Las dendritas llevan las entradas ponderadas (W*X) al cuerpo de la neurona artificial, donde son sumadas entre sí y con un sesgo b. Si esta suma sobrepasa un cierto umbral, que depende a su vez de la función de activación, entonces la neurona se dispara [39]. En la Figura 3.2 se presenta el esquema de una neurona artificial con 3 entradas y su modelo matemático.

Cabe señalar que la sumatoria de las entradas ponderadas (también llamadas activaciones) sólo forman parte de una componente lineal y que es la función de activación la que proporciona las cualidades no lineales a la neurona. Existen varios tipos de funciones de activación, entre las que destacan la sigmoidal, la tangente hiperbólica (tanh) y la unidad lineal rectificadora (ReLU) [46].

Las redes neuronales corresponden a conjuntos de neuronas que están conectadas en un grafo donde las salidas de algunas neuronas se convierten en entradas para otras neuronas. Existen diferentes arquitecturas, pero la más común es el perceptrón multicapa (MLP), el cual esta formado por capas totalmente conectadas (*Fully connected*, en inglés), en las que las neuronas de una capa están conectadas con todas y cada una de las neuronas de la capa contigua [39]. Un diagrama de MLP de 2 capas se muestra en la Figura 3.3, que cuenta con una capa oculta con 4 neuronas y una de salida con 2 neuronas (no se cuentan las entradas).

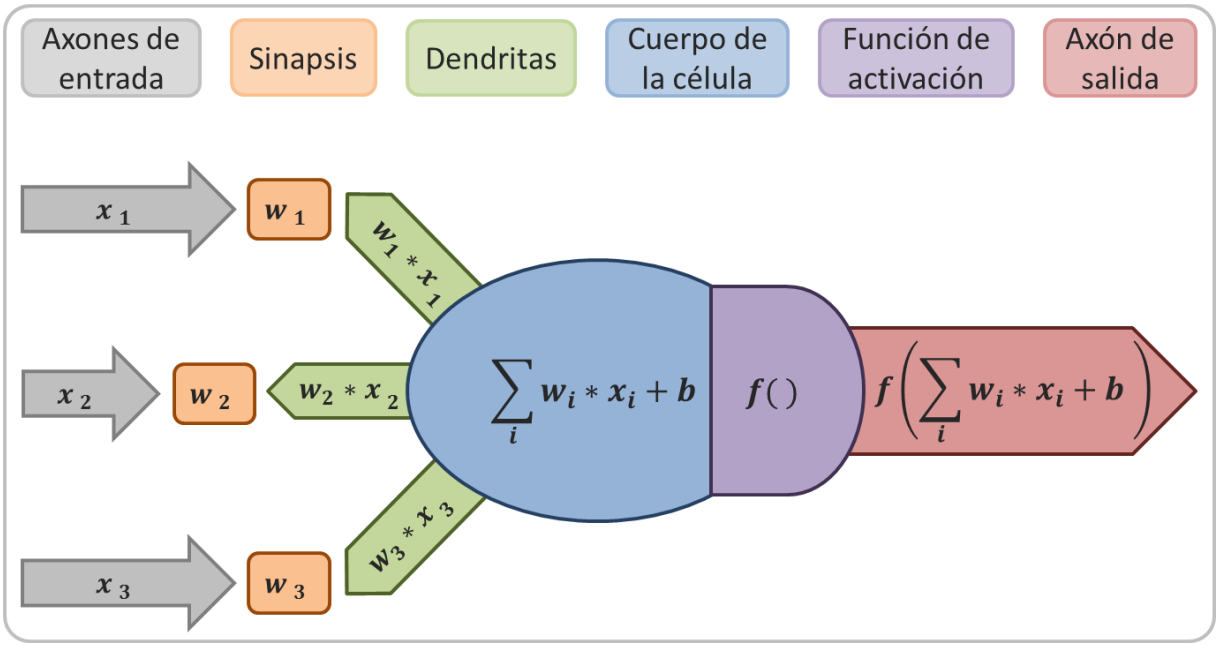

Figura 3.2 Esquema de una neurona artificial y su modelo matemático. Fuente: Elaboración propia

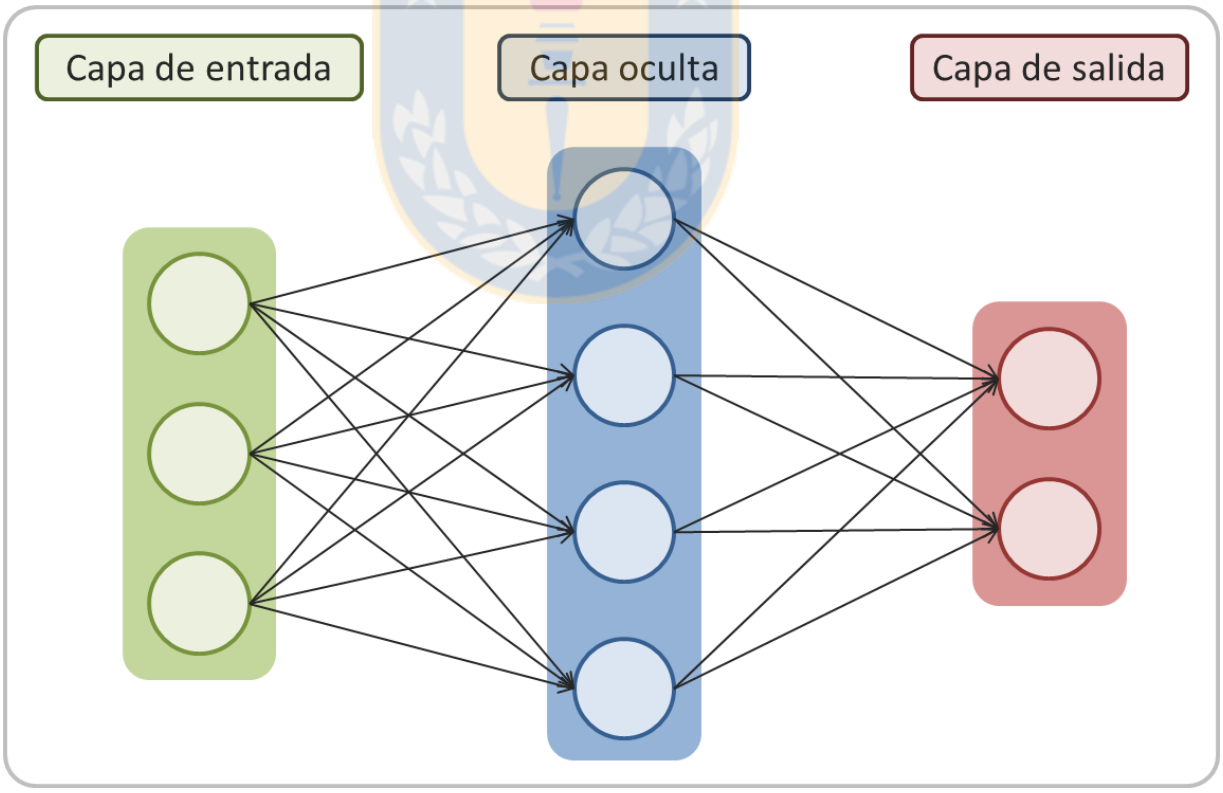

Figura 3.3 Diagrama de un perceptrón multicapa de 2 capas. Fuente: Elaboración propia

3.3.1 Unidad lineal rectificadora (ReLU)

La ReLU ha sido ampliamente usada en el último tiempo como función de activación en una neurona artificial [46]. Se calcula según la ecuación 3.5, que puede ser interpretada como establecer un umbral en cero para la activación de la neurona, tal como se ilustra en la Figura 3.4.

$$f(x) = \max(0, x) \tag{3.5}$$

Las redes neuronales que utilizan esta función muestran una convergencia del gradiente estocástico descendiente más rápido que las tradicionales sigmoidal y tanh, lo que se debe a que no se satura como las otras. Además, la ReLU es menos costosa computacionalmente, pues sólo aplica un umbral de cero a las activaciones en vez de computar varias operaciones matemáticas.

Sin embargo, la ReLU puede presentar problemas en el entrenamiento, pues es posible que los pesos de una neurona se actualicen de manera que nunca más vuelva a activarse. Esto se puede prevenir ajustando correctamente la tasa de aprendizaje.

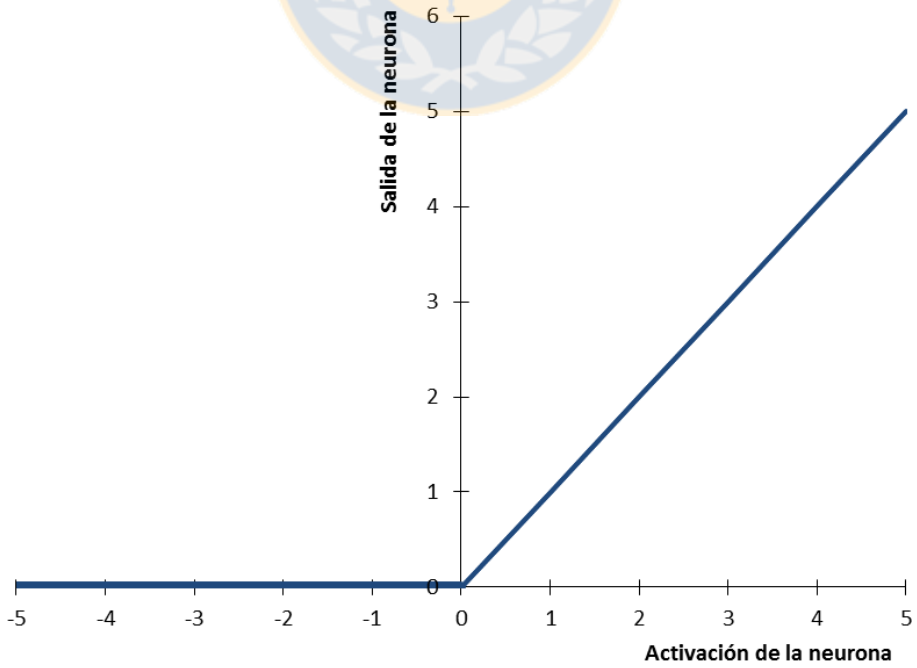

Figura 3.4 Representación gráfica de la función de activación ReLU. Fuente: Elaboración propia

3.3.2 Inicialización de variables

Antes de la etapa de entrenamiento en una red neuronal, es necesario inicializar los parámetros de la misma. La recomendación general para inicializar los pesos es que sean valores aleatorios pequeños, pero no cero, con distribución gausiana o uniforme de media cero y normalizando la varianza de la salida de cada neurona. En cuanto a los sesgos, se recomienda inicializarlos con un valor constante pequeño o igual a cero.

A medida que una red se hace más profunda, es más importante la decisión sobré qué tipo de inicialización se utilizará, por lo mismo se han realizado varios trabajos en esta línea. La inicialización de variables *Xavier*, explicado en el trabajo de Glorot y Bengio [47], busca que la varianza en cada capa de neuronas se mantenga igual (entrada = salida), de manera de evitar que la señal "explote" a un valor alto o se "desvanezca" en cero.

En esta misma línea, Ioffe y Szegedy [48], desarrollaron una técnica denominada normalización de paquetes (*batch normalization*, en inglés), que fuerza a que las activaciones de una red adopten una distribución gausiana unitaria al comienzo del entrenamiento; por lo anterior, se utiliza inmediatamente después de una capa totalmente conectada o de una convolución, y antes de una no linealidad (función de activación). Mediante la utilización de esta técnica, las redes poseen mayor robustez a una mala inicialización de parámetros, pues se podría interpretar como aplicar un preprocesamiento de los datos en cada capa.

3.3.3 Regularización

Se han desarrollado variadas técnicas con la finalidad de prevenir el sobre-ajuste de las redes neuronales artificiales. La más común de ellas es la regularización L2, que por cada peso w, se agrega el término $\frac{1}{2}\lambda w^2$ a la función objetivo, lo que podría ser interpretado como penalizar los pesos notoriamente elevados y preferir los pesos difusos [49].

Recientemente se ha introducido una técnica de regularización estocástica muy efectiva llamada *Dropout*, que durante el entrenamiento de una red neuronal mantiene activa una neurona con probabilidad p, o la fuerza a cero con probabilidad $1 - p$ [50]. Ver Figura 3.5.

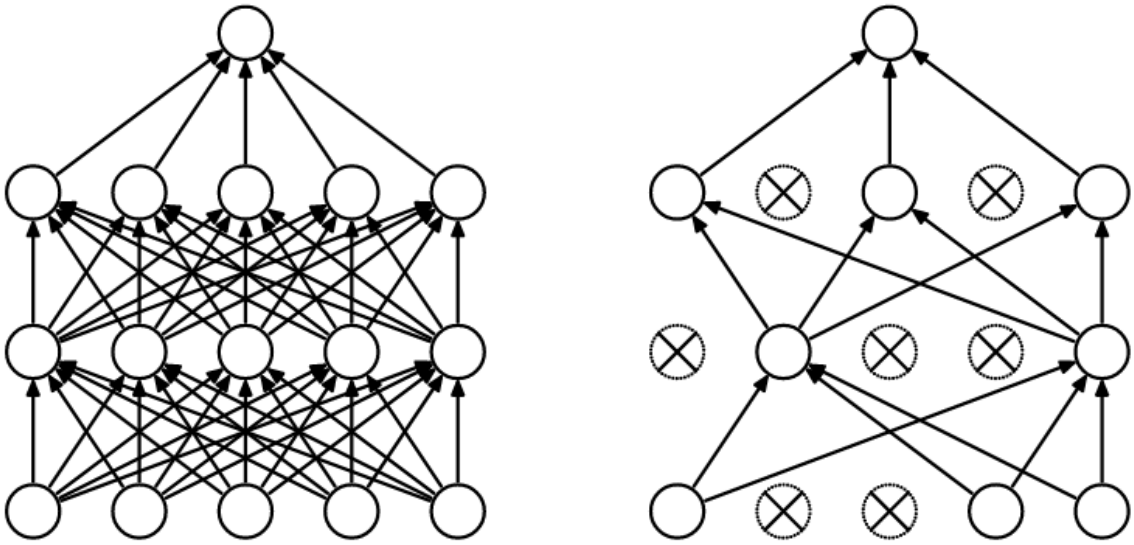

Figura 3.5 *Dropout* en una red neuronal. A la izquierda, una red neuronal estándar, a la derecha, la misma red luego de aplicar *dropout*. Las unidades con una cruz han sido inhabilitadas. Fuente: [50]

3.3.4 Clasificador *Softmax*

La función de pérdida o de costo es la función objetivo utilizada en la etapa de entrenamiento para ir ajustando los pesos del modelo, en base a la compatibilidad entre una predicción y el valor real de una variable. Para el problema de clasificación multiclase, en el que cada ejemplo posee una sola etiqueta correcta, se suele utilizar el clasificador *Softmax*, que usa a su vez, la pérdida de entropía cruzada (L_i) como función de pérdida [39].

Las activaciones de la capa de salida de una red neuronal se definen como $f = f(x_i; W)$, entonces cada ejemplo del vector f posee puntajes para cada clase. Considerando lo anterior, la pérdida de entropía cruzada se define según la ecuación (3.6).

$$L_i = -\ln\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{3.6}$$

El término contenido dentro del logaritmo natural es la función Softmax, que toma un vector de puntajes f ($\in \mathbb{Z}$) para cada clase j y lo traspasa (con exponenciación y normalización) a un vector con valores entre 0 y 1, que en total suma 1. El componente f_{y_i}

representa el vector de puntajes para la clase real correcta. Finalmente, se calcula la pérdida de entropía cruzada con la probabilidad normalizada de la clase real del ejemplo en evaluación.

Para comprender de mejor forma estos conceptos, se presenta un ejemplo en la Tabla 3.2, en el que se calcula la pérdida de entropía cruzada a partir de los puntajes de cada clase y considerando que la clase real correcta es 4.

Clase	Puntajes para cada clase	Probabilidades no normalizadas	Probabilidades normalizadas	Pérdida de entropía cruzada
	$f(x_i; W)$	$e^{f_{y_i}}$	$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$	$-\ln\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$
1	-3,44	0,0321	0,0006	-
2	1,16	3,1899	0,0596	-
3	-0,81	0,4449	0,0083	-
4	3,91	49,8990	0,9315	0,0709

Tabla 3.2 Ejemplo de cálculo de pérdida de entropía cruzada. Fuente: Elaboración propia

3.3.5 Entrenamiento de parámetros

La función de pérdida permite evaluar la calidad de un conjunto de pesos W, por lo que lo que se busca con el proceso de entrenamiento, es encontrar un conjunto de pesos que minimice la función de pérdida.

La forma más usada para optimizar la función de pérdida, es a través del cálculo de gradientes, que indican la dirección hacia donde hay una pendiente mas pronunciada, lo que es útil en la minimización de la pérdida. Los gradientes corresponden a una generalización del cálculo de la pendiente para una función, que en vez de recibir un único número, recibe un vector de números; así, un gradiente es un vector de pendientes (o derivadas). La derivada de una función con respecto a su entrada se define según la ecuación (3.7) y cuando la función considera un vector de números en vez de uno sólo, el gradiente es el vector de derivadas parciales en cada dimensión.

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.7}$$

El gradiente indica la dirección hacia la cual la función decrece con mayor intensidad, pero no entrega información sobre cuál debe ser la magnitud del movimiento, también referido como tasa de aprendizaje. Si es muy pequeña, se obtendría un progreso consistente, pero muy lento; si es muy grande, se obtendría un progreso más rápido, pero más riesgoso.

El procedimiento de evaluar el gradiente y actualizar los parámetros se denomina gradiente descendiente. En el caso que se realice este procedimiento por pequeños subconjuntos del conjunto de entrenamiento, se denomina gradiente descendiente por mini paquetes. El caso particular en el que se calculan los gradientes y se actualizan los parámetros de a un ejemplo a la vez, se le denomina gradiente descendiente estocástico o en línea [51].

Para calcular como los gradientes fluyen hacia atrás (desde la salida hacia la entrada), se utiliza el mecanismo denominado propagación hacia atrás (*backpropagation*, en inglés), que utiliza la regla de la cadena para ir calculando secuencialmente los gradientes individuales de cada uno de los parámetros [52]. Por ejemplo, si se tiene la función $f(x, y, z) = (x + y)z$ se podría reescribir como $f = uz$, considerando $u = x + y$. De esta forma, se podrían calcular los gradientes $\frac{\partial f}{\partial u} = z$ y $\frac{\partial f}{\partial z} = u$. Además se tiene que $\frac{\partial u}{\partial x} = 1$ y $\frac{\partial u}{\partial y} = 1$. Como lo que se quiere es representar el gradiente en función de las entradas, se utiliza la regla de la cadena para unir las expresiones anteriores a través de una multiplicación, otorgando como gradientes con respecto a x y a y las expresiones de las ecuaciones (3.8) y (3.9) respectivamente.

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} * \frac{\partial u}{\partial x} \tag{3.8}$$

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial u} * \frac{\partial u}{\partial y} \tag{3.9}$$

El mismo procedimiento aplicado en el ejemplo anterior se puede ir aplicando hacia atrás en una red de varias capas, por ejemplo, considerando la entrada x del ejemplo como el resultado de una función en la etapa anterior y calculando las derivadas parciales con respecto a sus entradas.

Una vez que los gradientes han sido calculados con propagación hacia atrás, se utilizan para ejecutar la actualización de los parámetros. Hay varias formas de llevarla a cabo, pero la recomendada por defecto en la actualidad, dado sus buenos resultados, es el algoritmo *Adam* [53], que corresponde a un método con tasa de aprendizaje adaptativo en función del concepto físico de momento. Los parámetros que los autores recomiendan por defecto son: $\varepsilon=10^{-8}$, $\beta_1=0{,}9$ y $\beta_2=0{,}999$.

Cada vez que un mini paquete de ejemplos es procesado por la red neuronal hacia adelante, computa el error, calcula los gradientes, propaga hacia atrás los gradientes y los aplica, entonces se dice que se ha cumplido una iteración. Cada vez que todo el conjunto de entrenamiento pasa por este proceso, se dice que se ha cumplido una época.

3.4. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son muy parecidas a las redes neuronales artificiales tradicionales, pues poseen neuronas con pesos y sesgos "aprendibles". Dado que están inspiradas en el funcionamiento de la corteza visual del cerebro, lo que caracteriza a las CNN es que consideran los datos de entrada como imágenes, logrando aplicar ciertas propiedades a la red que reducen la cantidad de parámetros y la hacen más eficiente de implementar. Las capas más utilizadas en la construcción de CNN son: convolucionales, de pooling y totalmente conectadas (ver sección 3.3). A continuación se detallan las primeras dos.

3.4.1 Capa convolucional

Una capa convolucional [54] consiste en un conjunto de filtros espaciales pequeños y "aprendibles", cada uno de los cuales es deslizado a través de los datos de entrada aplicando el producto punto entre los valores del filtro y los de la entrada en el área ocupada por el filtro; proceso denominado convolución. Estas activaciones se apilan a lo largo de la dimensión de profundidad y conforman el volumen de salida.

Para reducir el número de parámetros de una capa convolucional, se utiliza un esquema de pesos compartidos, que asume que si una característica es útil en una posición espacial, entonces también debe ser útil en otra posición espacial. Considerando lo anterior y que las

capas convolucionales van apilando las salidas en la dimensión de la profundidad, entonces se determina que las neuronas de cada “rebanada” (en la dimensión de profundidad) comparten los pesos W y los sesgos b, reduciendo considerablemente el número de parámetros a entrenar.

Los hiperparámetros involucrados en una capa convolucional son: número de filtros (o profundidad del volumen de salida), tamaño de los filtros, desplazamiento del filtro (*stride*) y forma de manejar los bordes (*padding*). El desplazamiento hace alusión a cuanto se debe desplazar el filtro en cada multiplicación con los datos de entrada, mientras que la forma de manejar los bordes se refiere a que es posible agregar ceros en los bordes con el fin de lograr que el tamaño de salida sea el mismo del de entrada (“Same”).

Para una mejor comprensión de esta capa, en la Figura 3.6 se despliega un esquema de ejemplo en el que se aplica un filtro de 3x3 a unos datos de entrada de 5x5, con desplazamiento 1 y sin manejo de bordes. Se muestra la forma de calcular la salida para los 2 primeros valores y luego se muestra el volumen de salida final calculado consecuentemente.

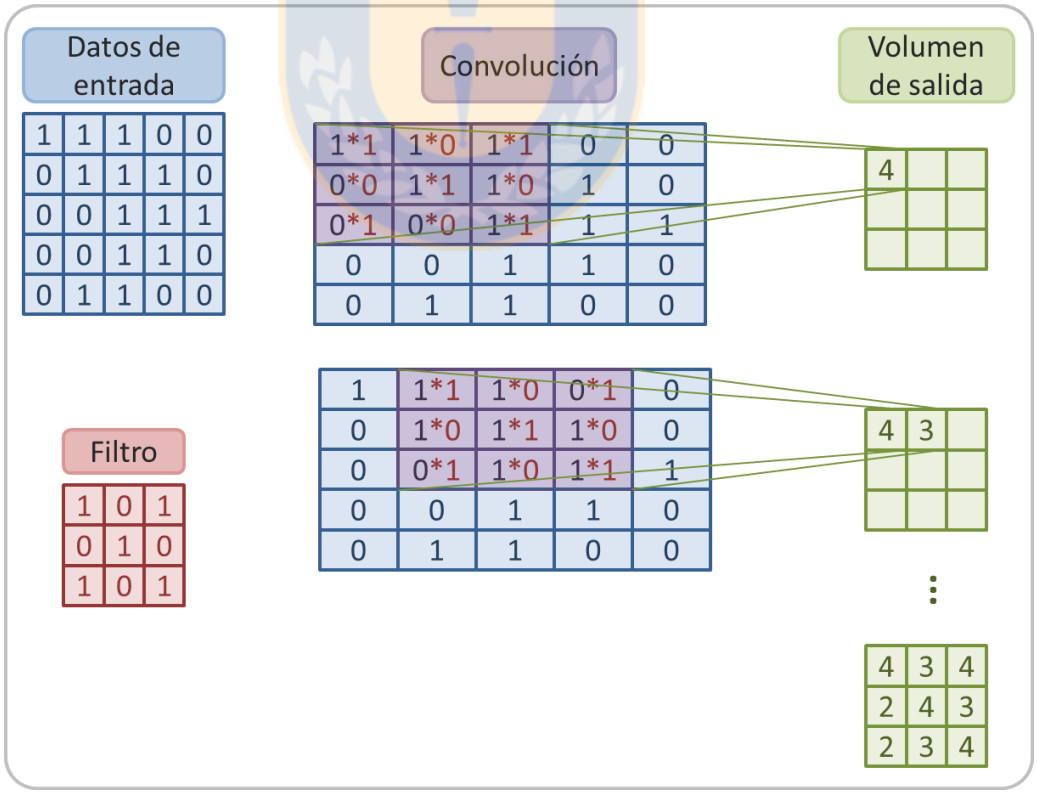

Figura 3.6 Esquema de una convolución. Fuente: Elaboración propia

3.4.2 Capa de *Pooling*

Esta capa [54] se utiliza con el fin de reducir la cantidad de parámetros y, por consiguiente, de procesamiento en una CNN; además de ayudar a controlar el sobre-ajuste. La capa de “pooling” opera en cada “rebanada” de la dimensión de profundidad de la entrada y la redimensiona espacialmente (sub-muestreo) utilizando una función.

El tipo de “pooling” más utilizado es el “max-pooling” que usa la función “max” y ha demostrado mejores resultados. Los hiperparámetros de esta capa son: tamaño del filtro y desplazamiento (*stride*).

En la Figura 3.7 se muestra un esquema del funcionamiento de una operación “max-pooling” de tamaño 2x2 y desplazamiento 2 en una “rebanada” de profundidad bidimensional.

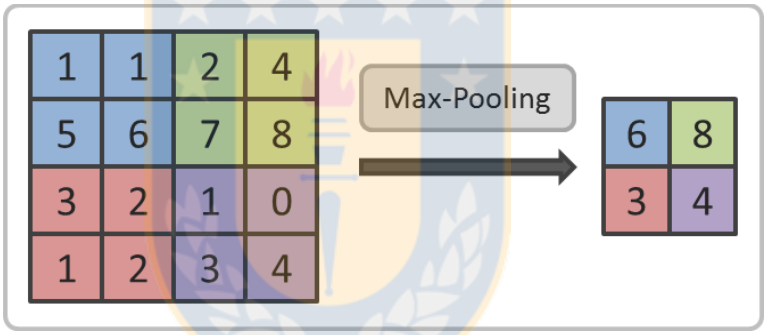

Figura 3.7 Esquema de una operación de "max-pooling". Fuente: Elaboración propia

3.5. Tensorflow

Tensorflow [55] es una librería de código abierto desarrollada por Google Brain Team dedicado al aprendizaje de máquina, en específico a la construcción y entrenamiento de redes neuronales. Esta librería maneja arreglos multidimensionales denominados tensores e interpreta los cálculos computacionales en forma de flujo de datos en grafos. Los algoritmos pueden ser implementados tanto en una unidad central de procesamiento (CPU) como en una unidad gráfica de procesamiento (GPU), asi como también en una unidad de procesamiento de tensores (TPU), dispositivo diseñado por Google específicamente para aprendizaje de máquina.

Capítulo 4. Metodología

La metodología usada en este trabajo se conforma de dos partes principales: la primera considera el procesamiento de la base de datos y, la segunda, la construcción de redes neuronales convolucionales. Ambas fueron implementadas en Python 2.7 [56] y utilizando un abanico de librerías de libre disposición.

4.1. Procesamiento de la base de datos

La base de datos (BD) utilizada para la construcción de instancias para entrenamiento y prueba de los modelos de clasificación fue la BD de arritmias del MIT-BIH [28]. Esta BD contiene 48 registros, cada uno de los cuales dispone de 3 archivos y que son descritos a continuación:

I. **Datos (".dat"):** Contiene las señales de ECG.
II. **Encabezado (".hea"):** Contiene información del registro, como: nombre del registro, N° de canales, largo de la señal, frecuencia de muestreo, resolución de la señal, nombre de las derivaciones de ECG, fármacos consumidos, entre otros.
III. **Anotaciones (".atr"):** Contiene la posición y las etiquetas de cada latido de *Physionet* [27].

Inicialmente, se implementó un algoritmo para descargar los 3 archivos de todos los registros de la BD, utilizando la librería *urllib*, cuyo pseudocódigo se muestra en la Tabla 4.1.

Registros = {lista con nombres de registros}
Link_base = "https://www.physionet.org/physiobank/database/mitdb/"
Por cada ***Nombre_registro*** en ***Registros***:
Por cada ***Extensión*** en {".dat", ".atr", ".hea"}:
Elemento = concatenar ***Nombre_registro*** con ***Extensión***
Link = concatenar ***Link_base*** con ***Elemento***
Descargar ***Link***
Fin de algoritmo

Tabla 4.1 Pseudocódigo para descargar BD. Fuente: Elaboración propia

Para interpretar, manipular y visualizar los registros, fue necesario usar la herramienta *wfdb* (desarrollada en C++), provista por *Physionet*, la cual puede ser operada a través de Python después de instalar los *wrappers* o traductores de código necesarios entre C++ y Python. La herramienta *wfdb* fue utilizada para abrir las señales (2 derivaciones de ECG por registro) y almacenarlas en un formato reconocible por Python, a través de la librería serial *cPickle*. Además, se utilizó la misma herramienta para interpretar las anotaciones de los registros, de esta manera, se logró cambiar las etiquetas originales para agruparlas en las 5 superclases propuestas por el estándar de la AAMI (Ver Sección 2.3); junto a ello, se almacenaron las posiciones todos los latidos (complejo QRS). El pseudocódigo del algoritmo para interpretar los archivos de los registros y almacenarlos en un formato reconocible por Python, se muestra en la Tabla 4.2.

Posteriormente, se procedió a atenuar dos tipos de ruido presentes en las señales de ECG de manera secuencial, utilizando las librerías de Python *numpy* y *scipy*:

I. **Desplazamiento de la línea base:** Se implementó un filtro de media móvil simple, con una ventana móvil de tamaño correspondiente al doble de la frecuencia de muestro de las señales y manejando los bordes de manera válida. Las señales resultantes, luego de aplicar la media móvil, fueron sustraídas de las originales, obteniendo señales sin variación de la línea base.

Registros = {lista con nombres de registros}
Por cada ***Nombre_registro*** en ***Registros***:
Canal1, ***Canal2*** = wfdb.pyrdsamp(***Nombre_registro***)
Anotaciones = wfdb.pyrdann(***Nombre_registro***)
Anotaciones_útiles = Sólo caracteres alfabéticos de ***Anotaciones***
Localizacion_Anotación = N° de muestra de cada elemento en ***Anotaciones***
Nombre_Señal = concatenar ***Nombre_registro*** con ".pkl"
Nombre_Anotación = concatenar ***Nombre_registro*** con "anot.pkl"
Guardar con cPickle los datos de ***Canal1*** y ***Canal2*** en el archivo ***Nombre_señal***
Guardar con cPickle los datos de ***Anotaciones_útiles*** y ***Localización_anotación*** en el archivo ***Nombre_Anotación***
Fin de algoritmo

Tabla 4.2 Pseudocódigo para interpretar los registros. Fuente: Elaboración propia

II. **Ruido electromagnético de la red eléctrica:** Los registros presentan ruido de 60[Hz], por lo que se implementó un filtro digital de respuesta al impulso finita (FIR) rechaza-banda con una ventana de tipo Kaiser, de orden 180 y con un rango de atenuación de frecuencias entre 59 y 61[Hz].

Luego de filtrar todos los registros, se pasó a la etapa de segmentación, que corresponde a la partición de la señal en trozos correspondientes a latidos individuales. Se utilizaron las posiciones de los complejos QRS almacenadas previamente y se segmentaron los latidos en trozos de 360 muestras (1[s]), con el complejo QRS en el centro.

Para implementar algoritmos de aprendizaje de máquina es necesario tener conjuntos de entrenamiento y prueba, evitando que los latidos de un mismo registro estén presentes en ambos conjuntos para evitar sesgos (esquema denominado inter-paciente). De Chazal et al. [17] recomienda separar los registros en dos conjuntos de similar número de latidos, con las clases repartidas de manera más equitativa posible y utilizando el esquema inter-paciente. Así mismo, y considerando que cada elemento (número) corresponde al nombre de un registro, los autores recomiendan separar los conjuntos de la siguiente manera:

I. **Conjunto de entrenamiento:** 101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223 y 230.
II. **Conjunto de prueba:** 100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233 y 234.
III. **Registros excluidos:** 102, 104, 107 y 217. Debido a que presentan latidos con marcapasos, se recomienda excluirlos para no afectar el entrenamiento de los modelos.

Para realizar comparaciones justas con otras técnicas de clasificación de latidos presentes en la literatura, se adoptan las recomendaciones anteriores.

Se desarrolló un algoritmo para establecer la estructura de los datos de cada conjunto como un arreglo de 3 dimensiones. Donde, la primera representa el número de latidos. La segunda, las muestras de cada latido. La tercera, el número de canales (2 en todas las señales). Además, se incluye un arreglo unidimensional que contiene las etiquetas con la clase de cada

latido codificada en valores numéricos, como se muestra en la Tabla 4.3; cabe señalar que el significado de las clases se encuentra en la Tabla 2.1.

La estructura de datos antes mencionada se esquematiza en la Figura 4.1, donde se muestran los 2 canales, cada uno de los latidos en las filas, cada una de las muestras en las columnas y, por otro lado, el vector de clases que contiene la etiqueta de cada latido. El pseudocódigo del algoritmo que construye los conjuntos de entrenamiento y prueba con esta estructura de datos se muestra en la Tabla 4.4.

Superclase	Clases	Codificación
N	N, L, R, e, j	0
S	A, a, J, S	1
V	V, E	2
F	F	3
Q	P, f, Q	4

Tabla 4.3 Codificación de clases de latidos. Fuente: Elaboración propia

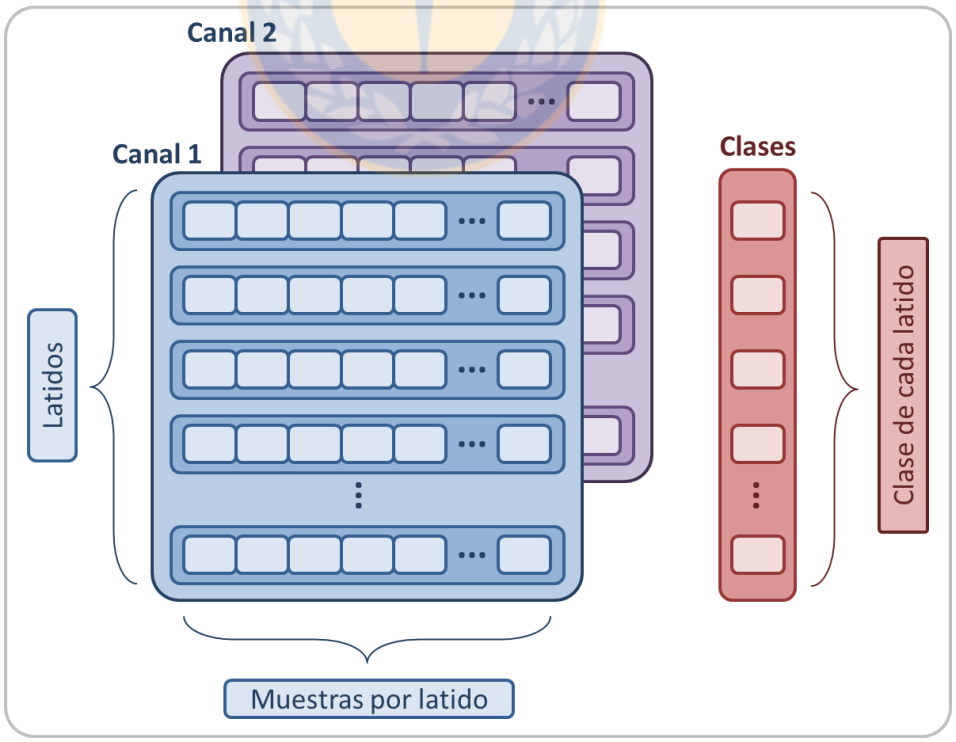

Figura 4.1 Estructura de datos para los conjuntos de entrenamiento y prueba. Fuente: Elaboración propia

Registros = {lista con nombres de registros de entrenamiento o prueba}
Ventana = Frecuencia de muestreo = 360
Características = Arreglo de 3 dimensiones
Clases = Arreglo de 1 dimensión
Por cada ***Nombre_registro*** en ***Registros***:
Canal1, ***Canal2*** = señales de ***Nombre_registro***
Loc_Anotación, ***Anotaciones*** = Cargar con cPickle las anotaciones de ***Nombre_registro***
Por cada ***posición*** en ***Loc_anotación:***
Latido1 = ***canal1 [posición-Ventana/2 : posición+Ventana/2]***
Latido2 = ***canal2 [posición-Ventana/2 : posición+Ventana/2]***
Latido_junto = apilar en el último eje ***Latido1*** y ***Latido2***
Etiqueta = ***Anotaciones [localización]***
Etiqueta_codificada = Codificar ***Etiqueta***
Apilar verticalmente ***Latido_junto*** en ***Características***
Apilar verticalmente ***Etiqueta_codificada*** en ***Clases***
Conjunto = apilar horizontalmente ***Clases*** y ***Características***
Guardar con cPickle los datos de ***Conjunto*** en el archivo ***Conjunto de {entrenamiento, prueba}***
Fin de algoritmo

Tabla 4.4 Pseudocódigo para construir conjuntos de entrenamiento y prueba. Fuente: Elaboración propia

4.2. Redes neuronales convolucionales

Las CNN implementadas constan de varias etapas modulares que pueden ir apilándose secuencialmente para construir modelos más o menos profundos. La librería escogida para la implementación de los modelos fue *Tensorflow* versión 0.12.

Se diseñaron e implementaron 4 diferentes estructuras de CNN, denominadas *CNN1F*, *CNN2F*, *CNN3F* y *CNN4F*, las cuales se detallan respectivamente en la Figura 4.2 (a), (b), (c) y (d). Todas estás estructuras poseen etapas secuenciales desde la entrada de los datos hasta la salida de los modelos, pasando por diferentes capas explicadas en la secciones 3.3 y 3.4.

Los hiperparámetros para cada estructura fueron determinados experimentalmente, siguiendo algunas de las recomendaciones de Bengio [57], y los definitivos se detallan en las Tablas 4.5, 4.6, 4.7 y 4.8. Cabe señalar que en la etapa de *reshape* el -1 significa que el largo en esa dimensión es variable. Además, se escogió la ReLU como función de activación.

En todas las CNN, los datos entrantes a la red pasan inicialmente por una etapa que los transforma en tensor, con la primera dimensión representando el número de latidos, la segunda, el número de muestras por latido (360 muestras), la tercera, la dimensión de altura (siempre 1) y la cuarta, el número de canales (2 canales).

Se inicializaron los pesos de las CNN implementadas con el algoritmo *Xavier* [47] y se entrenaron los parámetros con el algoritmo *Adam* [53], considerando la pérdida de entropía cruzada como función de costo. Además, se utilizó *dropout* en las capas totalmente conectadas [50]. Estos conceptos están explicados en la sección 3.3.

Durante el proceso de entrenamiento, en cada época se calculó la exactitud de entrenamiento y el tiempo transcurrido y, al completarse el número de épocas estipulado, se calculó: la exactitud de prueba, la matriz de confusión, el número de variables entrenables y el tiempo computacional utilizado (mediante la librería *timeit*). Además, se utilizó la herramienta *Tensorboard* de *Tensorflow* para registrar información sobre el proceso de entrenamiento, como la evolución de la función de pérdida y de la exactitud de entrenamiento.

Finalmente, teniendo la matriz de confusión se calcularon otros indicadores de rendimiento como: sensibilidad (Se), predictividad positiva (P^+) y tasa de falsos positivos (FPR); calculados según las ecuaciones de la sección 3.1. De esta manera, es posible comparar los modelos implementados con otros métodos disponibles en la literatura.

Nombre	**Capa**	**Hiperparámetros**		
CNN1F	Reshape	[-1, Frec. De muestreo, 1, N° de canales]		
	Conv1	N° de filtros: 25	Tamaño de filtros: 3	Bordes: “Same”
	BN1	-		
	Conv2	N° de filtros: 15	Tamaño de filtros: 6	Bordes: “Same”
	BN2	-		
	FC1	N° de neuronas: 75		
	FC2	N° de neuronas: 5 (N° de clases)		
	Softmax	-		
	Épocas	3		
	Tasa de aprendizaje	2×10^{-5}		
	Tamaño de paquetes	40		
	Dropout	0,5		

Tabla 4.5 Hiperparámetros de la red CNN1F. Fuente: Elaboración propia

Nombre	Capa	Hiperparámetros		
CNN2F	Reshape	[-1, Frec. De muestreo, 1, N° de canales]		
	Conv1	N° de filtros: 20	Tamaño de filtros: 3	Bordes: “Same”
	BN1	-		
	Pool1	Tamaño de pooling: 2	Desplazamiento: 2	Bordes: “Same”
	Conv2	N° de filtros: 15	Tamaño de filtros: 3	Bordes: “Same”
	BN2			
	Pool2	Tamaño de pooling: 2	Desplazamiento: 2	Bordes: “Same”
	Conv3	N° de filtros: 10	Tamaño de filtros: 3	Bordes: “Same”
	BN3	-		
	FC1	N° de neuronas: 75		
	FC2	N° de neuronas: 5 (N° de clases)		
	Softmax	-		
	Épocas	4		
	Tasa de aprendizaje	2×10^{-5}		
	Tamaño de paquetes	40		
	Dropout	0,5		

Tabla 4.6 Hiperparámetros de la red CNN2F. Fuente: Elaboración propia

Nombre	Capa	Hiperparámetros		
CNN3F	Reshape	[-1, Frec. De muestreo, 1, N° de canales]		
	Conv1	N° de filtros: 20	Tamaño de filtros: 10	Bordes: “Same”
	BN1	-		
	Conv2	N° de filtros: 10	Tamaño de filtros: 3	Bordes: “Same”
	BN2	-		
	Pool1	Tamaño de pooling: 3	Desplazamiento: 3	Bordes: “Same”
	Conv3	N° de filtros: 15	Tamaño de filtros: 6	Bordes: “Same”
	BN3	-		
	Pool2	Tamaño de pooling: 2	Desplazamiento: 2	Bordes: “Same”
	Conv4	N° de filtros: 15	Tamaño de filtros: 3	Bordes: “Same”
	BN4	-		
	Conv5	N° de filtros: 10	Tamaño de filtros: 3	Bordes: “Same”
	BN5	-		
	FC1	N° de neuronas: 80		
	FC2	N° de neuronas: 40		
	FC3	N° de neuronas: 5 (N° de clases)		
	Softmax	-		
	Épocas	9		
	Tasa de aprendizaje	2×10^{-5}		
	Tamaño de paquetes	25		
	Dropout	0,5		

Tabla 4.7 Hiperparámetros de la red CNN3F. Fuente: Elaboración propia

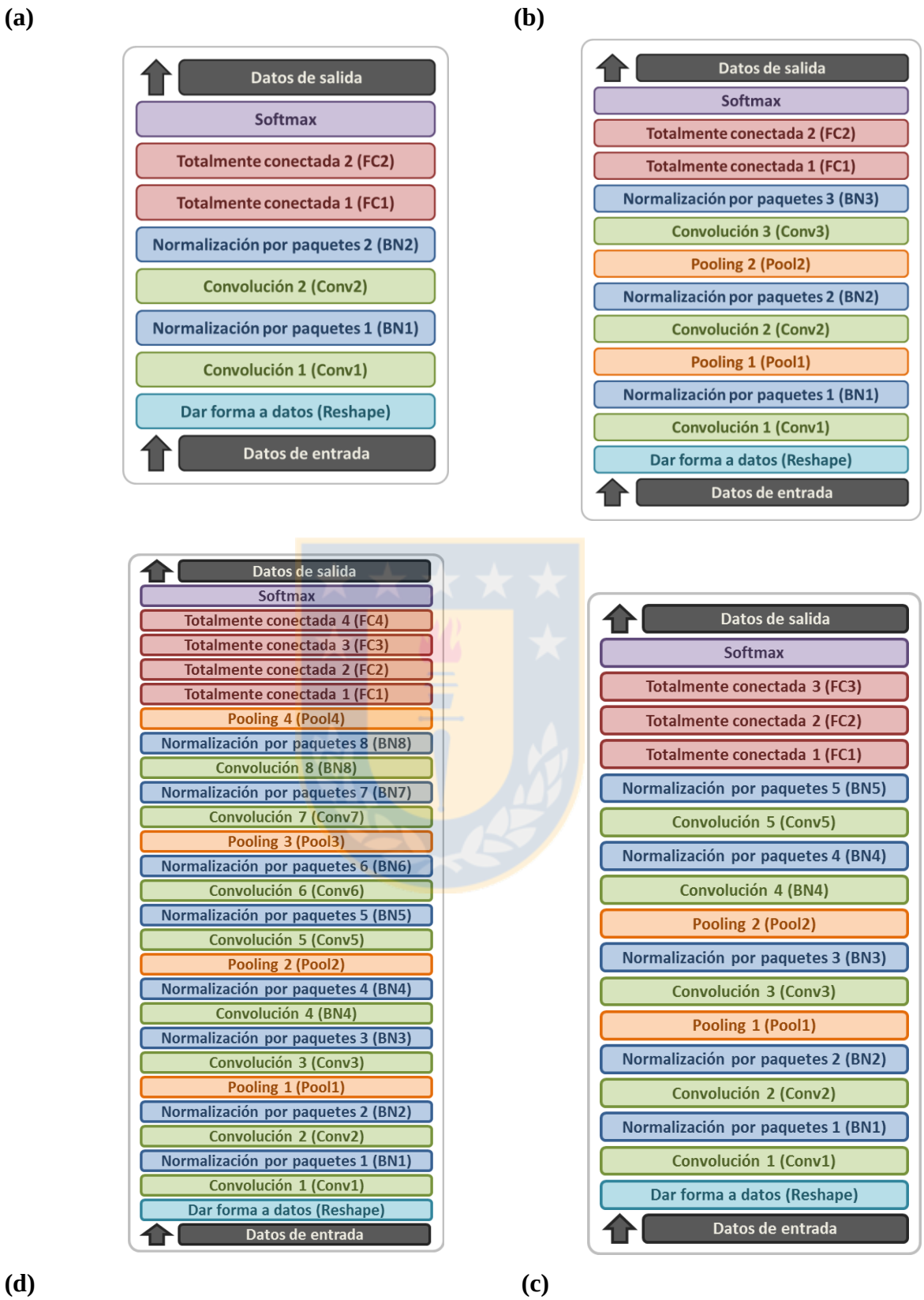

Figura 4.2 Estructura de las redes (a) CNN1F, (b) CNN2F, (c) CNN3F, (d) CNN4F. Para detalles sobre las capas referirse a las secciones 3.3 y 3.4. Fuente: Elaboración propia

Nombre	Capa	Hiperparámetros		
CNN4F	Reshape	[-1, Frec. De muestreo, 1, N° de canales]		
	Conv1	N° de filtros: 20	Tamaño de filtros: 10	Bordes: “Same”
	BN1	-		
	Conv2	N° de filtros: 10	Tamaño de filtros: 5	Bordes: “Same”
	BN2	-		
	Pool1	Tamaño de pooling: 4	Desplazamiento: 4	Bordes: “Same”
	Conv3	N° de filtros: 15	Tamaño de filtros: 4	Bordes: “Same”
	BN3	-		
	Conv4	N° de filtros: 10	Tamaño de filtros: 3	Bordes: “Same”
	BN4	-		
	Pool2	Tamaño de pooling: 3	Desplazamiento: 3	Bordes: “Same”
	Conv5	N° de filtros: 10	Tamaño de filtros: 5	Bordes: “Same”
	BN5	-		
	Conv6	N° de filtros: 8	Tamaño de filtros: 3	Bordes: “Same”
	BN6	-		
	Pool3	Tamaño de pooling: 2	Desplazamiento: 2	Bordes: “Same”
	Conv7	N° de filtros: 5	Tamaño de filtros: 3	Bordes: “Same”
	BN7	-		
	Conv8	N° de filtros: 5	Tamaño de filtros: 3	Bordes: “Same”
	Pool4	Tamaño de pooling: 2	Desplazamiento: 2	Bordes: “Same”
	BN8	-		
	FC1	N° de neuronas: 64		
	FC2	N° de neuronas: 48		
	FC3	N° de neuronas: 32		
	FC4	N° de neuronas: 5 (N° de clases)		
	Softmax	-		
	Épocas	9		
	Tasa de aprendizaje	2×10^{-5}		
	Tamaño de paquetes	25		
	Dropout	0,5		

Tabla 4.8 Hiperparámetros de la red CNN4F. Fuente: Elaboración propia

Capítulo 5. Resultados

Para implementar los algoritmos, se dispuso de un computador con procesador Intel Core i5-4200M a 2,5GHz y 4 núcleos, con 12GB de RAM y sistema operativo Windows 10 Pro de 64 bits. Sin embargo, las librerías necesarias no tenían compatibilidad con Windows, en especial *wfdb* y *Tensorflow*, por lo que se instaló el sistema operativo Ubuntu 16.04 LTS de 64bits en una máquina virtual VMWare Workstation 12 Player, otorgándole 9,4GB de RAM.

En Ubuntu, se creó un ambiente virtual llamado "tensorflow" y en él se instaló una distribución de Python (versión 2.7) especializada en la ciencia de datos, denominada Anaconda (versión 4.3), instalando a su vez, todas las librerías incluidas en el paquete estándar. Luego se instalaron otras no incluidas a través del instalador de Anaconda, denominado "Conda".

Por otro lado, en el mismo ambiente virtual nombrado anteriormente, se instalaron las librerías *wfdb 10.5.24* y *Tensorflow 0.12*, con todas sus dependencias.

Los resultados se presentan de la misma forma que la metodología, dividiéndolos en 2 partes, primero los relativos al procesamiento de la base de datos y, posteriormente, la construcción de redes neuronales convolucionales. En las siguientes secciones se detalla cada una de estas partes.

5.1. Procesamiento de la base de datos

En la Figura 5.1 se muestra un segmento de 10[s] del registro 101, con evidencia de un claro desplazamiento de línea base en sus 2 canales, correspondientes a: Segunda derivación de las extremidades modificada (MLII) y Quinta derivación precordial (V5), respectivamente. En el eje horizontal se muestra el N° de muestras, que considerando una frecuencia de muestreo de 360[Hz], se obtiene 1[s] de señal cada 360 muestras. En el eje vertical se muestra la amplitud en unidades de convertidor analógico-digital (ADC) codificado en 11bits (valores enteros posibles entre 0 y 2^{11}-1).

En cuanto a los filtros implementados para la atenuación de ruidos, se presentan los resultados considerando como ejemplo el canal 1 del registro 101. En las Figuras 5.2 y 5.3 se muestran los resultados en el dominio del tiempo y de la frecuencia para la señal con y sin desplazamiento de línea base, respectivamente. En la Figura 5.3, en el dominio del tiempo, se nota que luego de aplicar el filtro, el desplazamiento de la línea base se elimina y se aprecia la señal centrada en cero. En cuanto al dominio de la frecuencia, que se obtiene aplicando la transformada rápida de Fourier (FFT) [58], se observa que en la Figura 5.2 existe un componente notorio en las frecuencias bajas (cercanas a 0[Hz]) y que al aplicar el filtro es atenuado y se observa con menor ganancia en la Figura 5.3.

El resultado de aplicar secuencialmente el filtro para el ruido de la red eléctrica se muestra en la Figura 5.4, donde se observa en el dominio del tiempo que la señal se ve más "suavizada". En cuanto al dominio de la frecuencia, se observa que en la Figura 5.3 existe un componente notorio en los 60[Hz] que al aplicar el filtro es atenuado y ya no se observa. La respuesta en frecuencia para el filtro se muestra en la Figura 5.5, donde se observa una ganancia menor a 0,1 para 60[Hz].

Luego de filtrar todos los registros, se pasó a la etapa de segmentación, donde se dividen las señales en latidos individuales de 360 muestras (vectores) con el complejo QRS en el centro, como el que se muestra en la Figura 5.6.

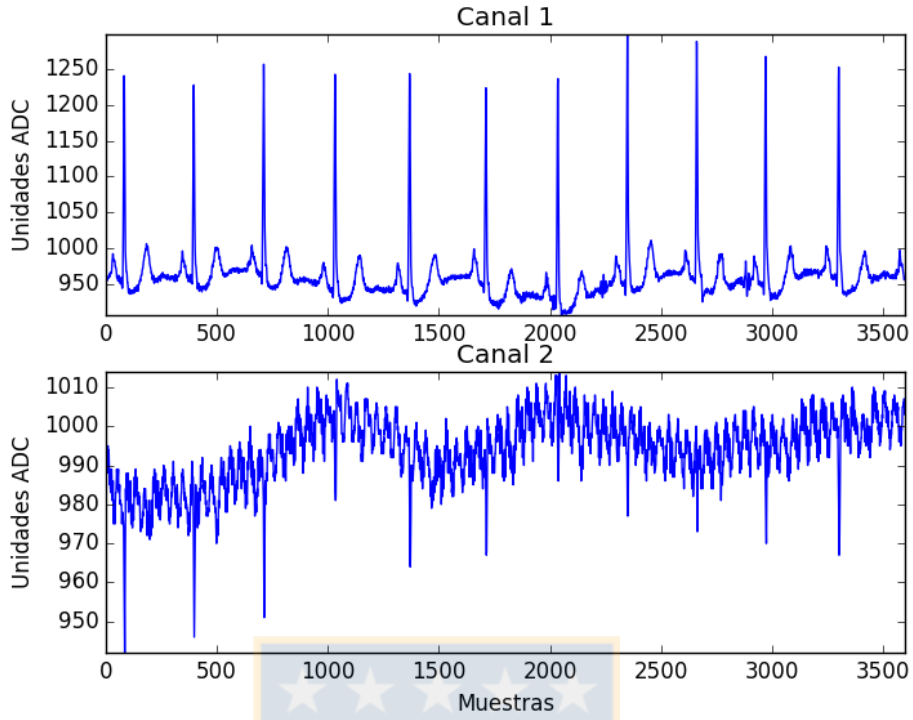

Figura 5.1 Segmento de 10[s] del registro 101. Fuente: Elaboración propia

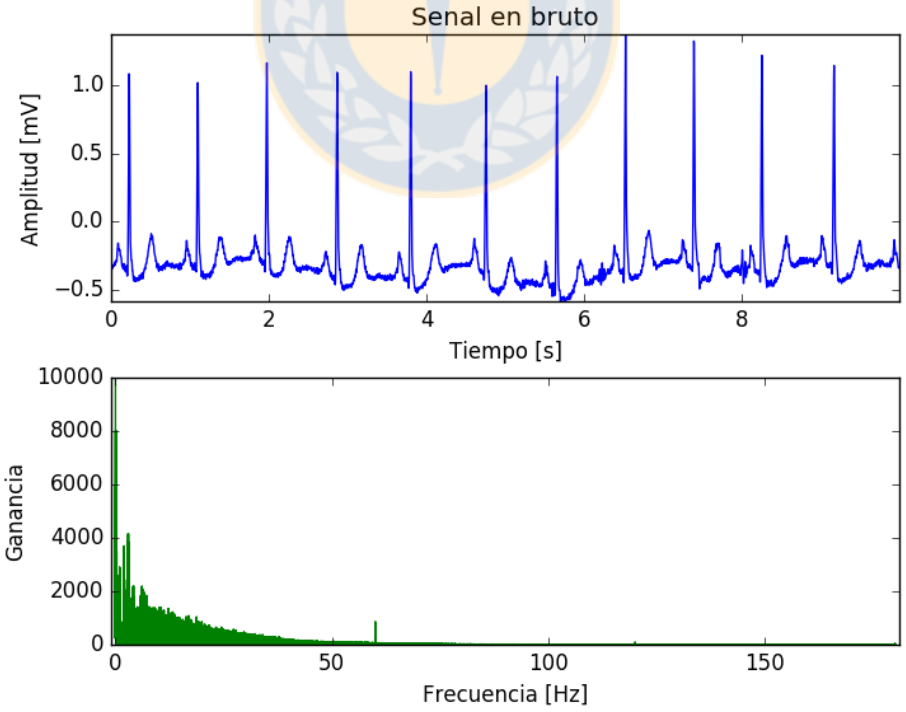

Figura 5.2 Registro 101 en dominio del tiempo (segmento de 10[s]) y dominio de la frecuencia. Fuente: Elaboración propia

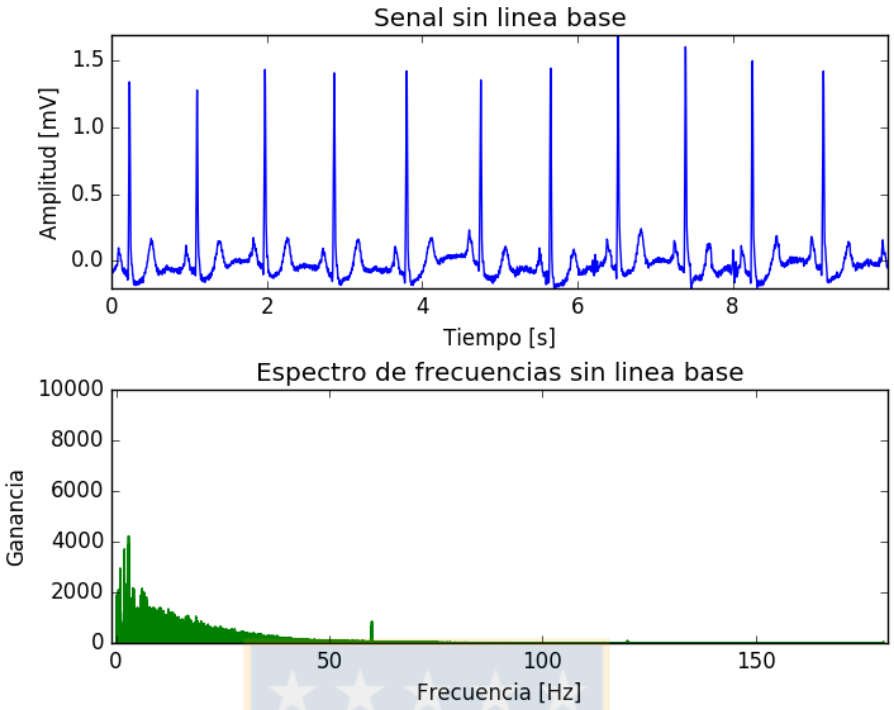

Figura 5.3 Registro 101 al aplicar filtro de línea base en dominio del tiempo y dominio de la frecuencia. Fuente: Elaboración propia

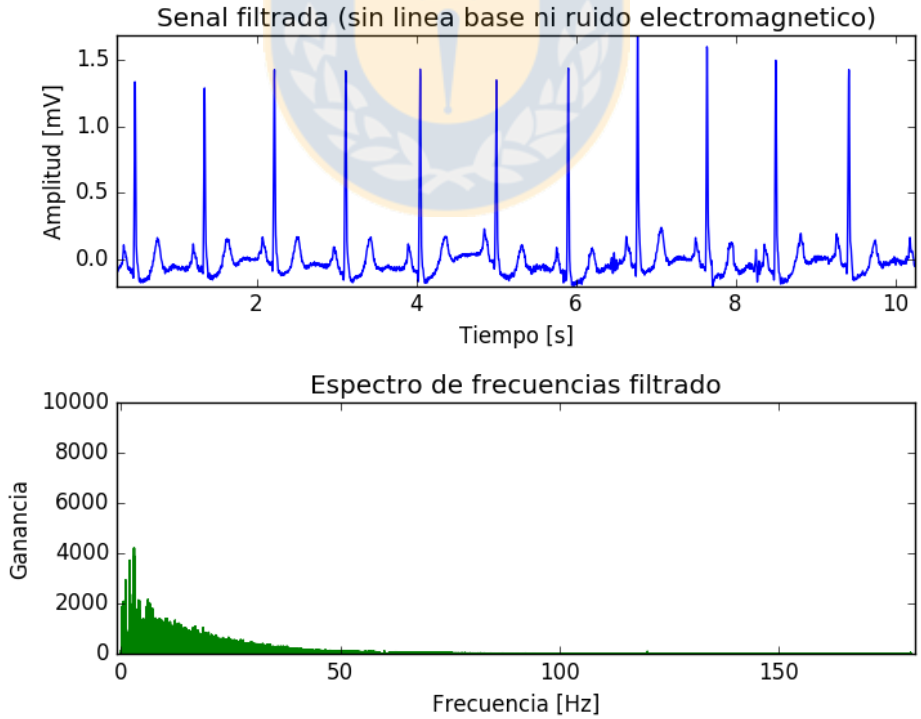

Figura 5.4 Registro 101 al aplicar secuencialmente ambos filtros señalados en dominio del y dominio de la frecuencia. Fuente: Elaboración propia

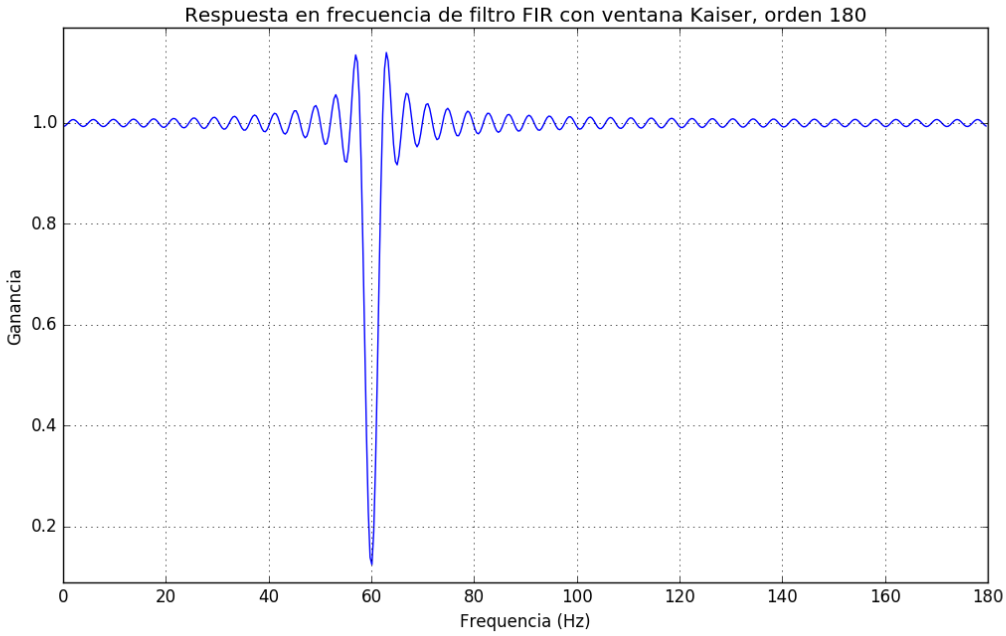

Figura 5.5 Respuesta en frecuencia del filtro digital implementado. Fuente: Elaboración propia

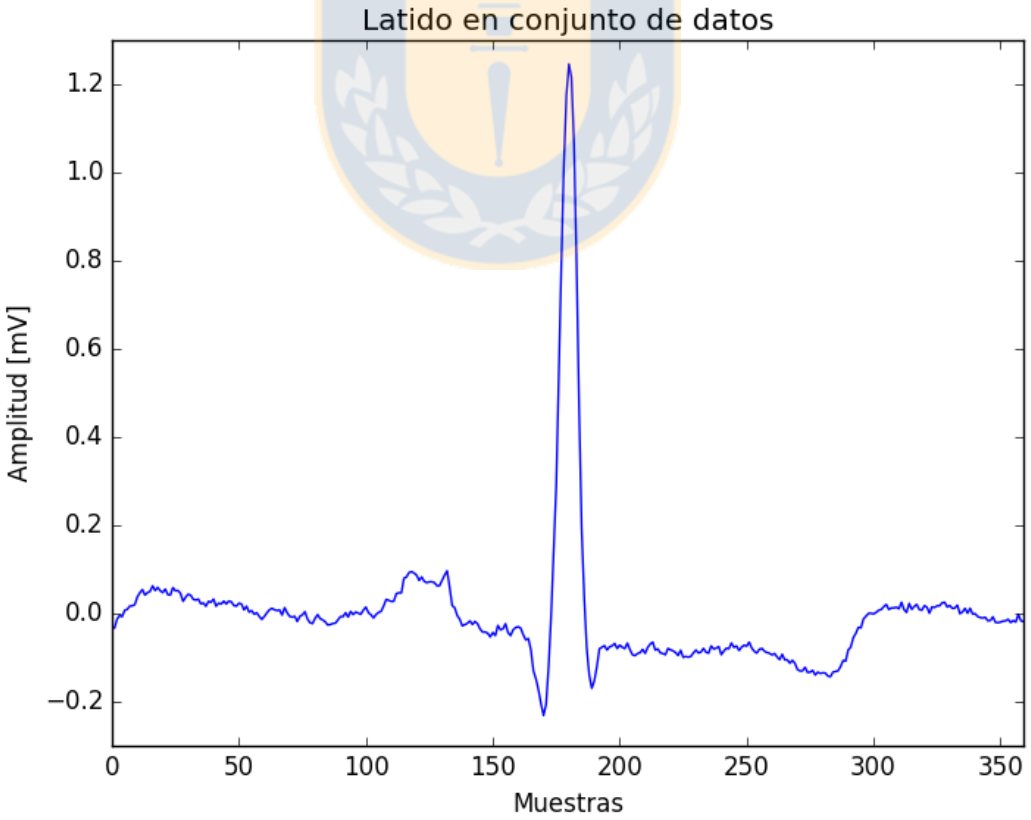

Figura 5.6 Latido de ejemplo del conjunto de datos (entrenamiento y prueba). Fuente: Elaboración propia

5.2. Redes neuronales convolucionales

Una vez que se ha realizado el procesamiento, ya se cuenta con los conjuntos de datos de entrenamiento y prueba, por lo que se procede a implementar las diferentes estructuras de CNN. Para cada una de ellas se despliega gráficamente la estructura en forma de grafo y se presenta la matriz de confusión y los indicadores de desempeño mencionados en la metodología. Además, mediante la herramienta *Tensorboard*, se exhibe el progreso del entrenamiento en cuanto a exactitud y función de pérdida.

En las secciones a continuación, se presentan los resultados obtenidos para cada una de las estructuras de CNN implementadas. Es de especial importancia el resultado de cada red en relación a la exactitud, pues es el indicador principal de desempeño de los modelos de clasificación.

5.2.1 CNN1F

Para la red CNN1F se muestra en la Figura 5.7 la estructura en forma de grafo obtenida de *Tensorboard*. Al igual que la estructura presentada en la metodología, los datos de entrada ingresan a la etapa de "dar forma a los datos" (abajo en la Figura 5.7), se va procesando a través de las diferentes capas, para finalmente entregar una clase en el clasificador *Softmax*. Luego de esto, se procede a las etapas de "Calcular matriz de confusión" y "Calcular exactitud". Además, se desprenden de las capas algunas variables como: los pesos W de las capas convolucionales y totalmente conectadas, "keep_prob" que contiene la probabilidad usada para la técnica de *dropout* y "BN_train" que es una variable booleana (binaria) que sirve para activar (conj. de entrenamiento) o desactivar (conj. de prueba) la normalización de paquetes.

La matriz de confusión y los indicadores de desempeño (ver sección 3.1) de la red CNN1F en la predicción del conjunto de pruebas, se muestran en las Tablas 5.1 y 5.2, respectivamente. La evolución de la exactitud y de la función de pérdida durante el entrenamiento se muestra en las Figuras 5.8 y 5.9, cuyo gráfico suavizado (naranjo) se despliega en función del número de iteraciones alcanzado. El color naranjo suave muestra los datos sin suavizar.

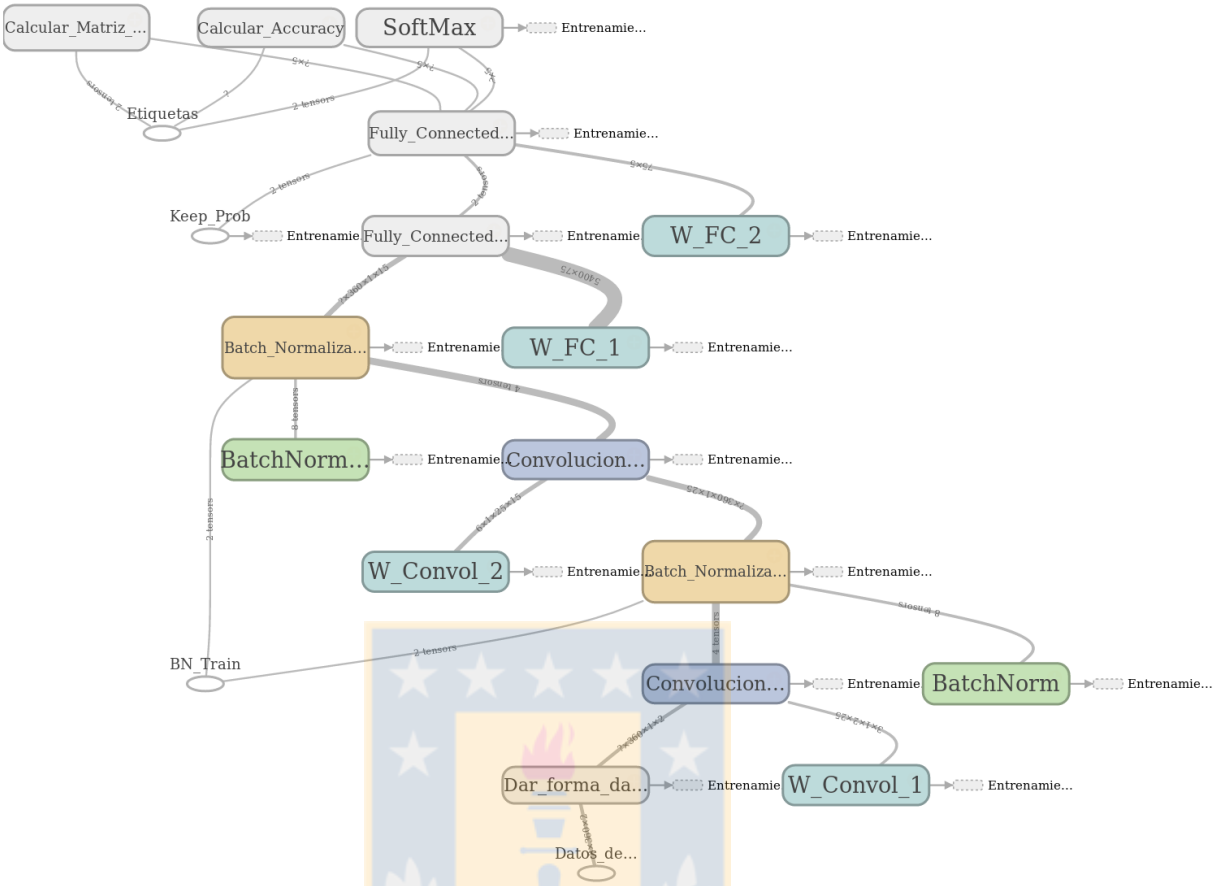

Figura 5.7 Grafo de la red CNN1F. Fuente: Elaboración propia

		Clases predichas por el modelo					Total
		N	S	V	F	Q	
Clases reales	N	43.635	388	206	2	0	44.231
	S	1.824	7	6	0	0	1.837
	V	585	9	2.572	54	0	3.220
	F	252	15	17	104	0	388
	Q	3	0	4	0	0	7
Total		46.299	419	2.805	160	0	49.683

Tabla 5.1 Matriz de confusión para red CNN1F en conjunto de prueba. Fuente: Elaboración propia

Indicador de desempeño	Clase	Resultado
Exactitud global (Acc)	Todas	0,9323
Sensibilidad (Se)	N	0,9865
	S	0,0038
	V	0,7988
	F	0,2680
	Q	0,0000
Predictividad positiva (P+)	N	0,9425
	S	0,0167
	V	0,9169
	F	0,6500
	Q	-
Tasa de falsos positivos (FPR)	N	0,4886
	S	0,0086
	V	0,0050
	F	0,0011
	Q	0,0000
N° de variables entrenables	-	407.935
Tiempo de procesamiento [s]	-	2186

Tabla 5.2 Indicadores de desempeño de red CNN1F en conjunto de prueba. Fuente: Elaboración propia

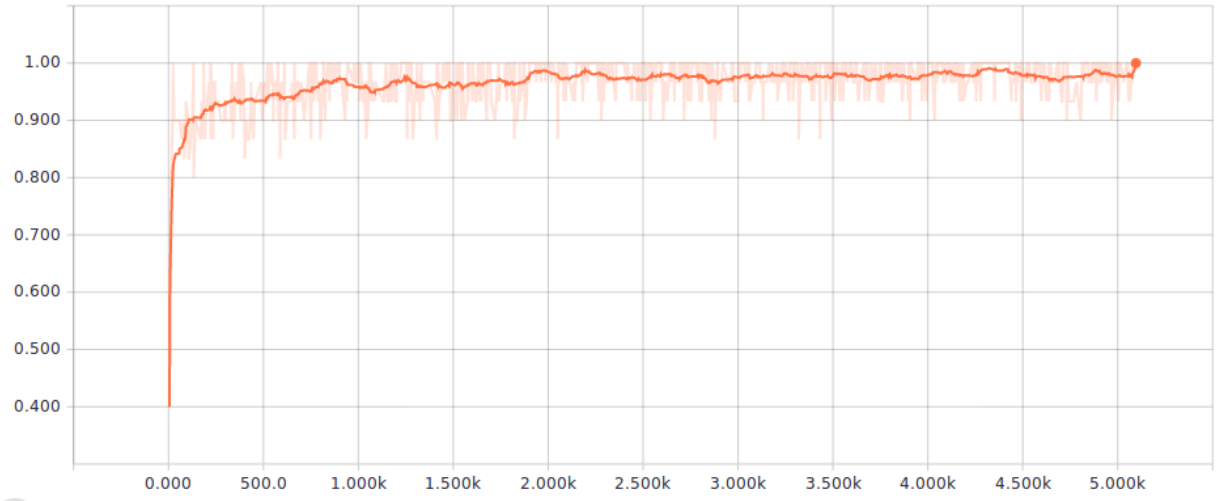

Figura 5.8 Exactitud en fase de entrenamiento de CNN1F. Eje X: N° de iteraciones. Eje Y: Exactitud (Acc). Fuente: Elaboración propia

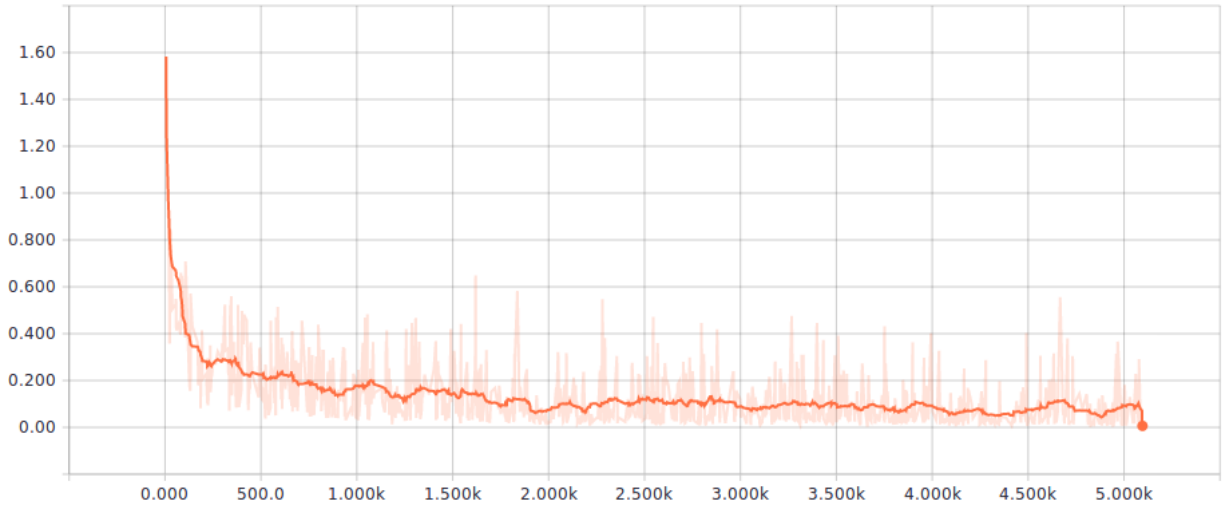

Figura 5.9 Función de pérdida en fase de entrenamiento de CNN1F. Eje X: N° de iteraciones. Eje Y: Pérdida de entropía cruzada. Fuente: Elaboración propia

5.2.2 CNN2F

El grafo de la red CNN2F se muestra en la Figura 5.10 y su descripción es la misma de la Figura 5.7, pero considerando que existen más capas. La matriz de confusión y los indicadores de desempeño de la red CNN2F, se muestran en las Tablas 5.3 y 5.4, respectivamente. La evolución de la exactitud y de la función de pérdida durante el entrenamiento se muestra en las Figuras 5.11 y 5.12, cuyo gráfico se despliega en función del número de iteraciones alcanzado.

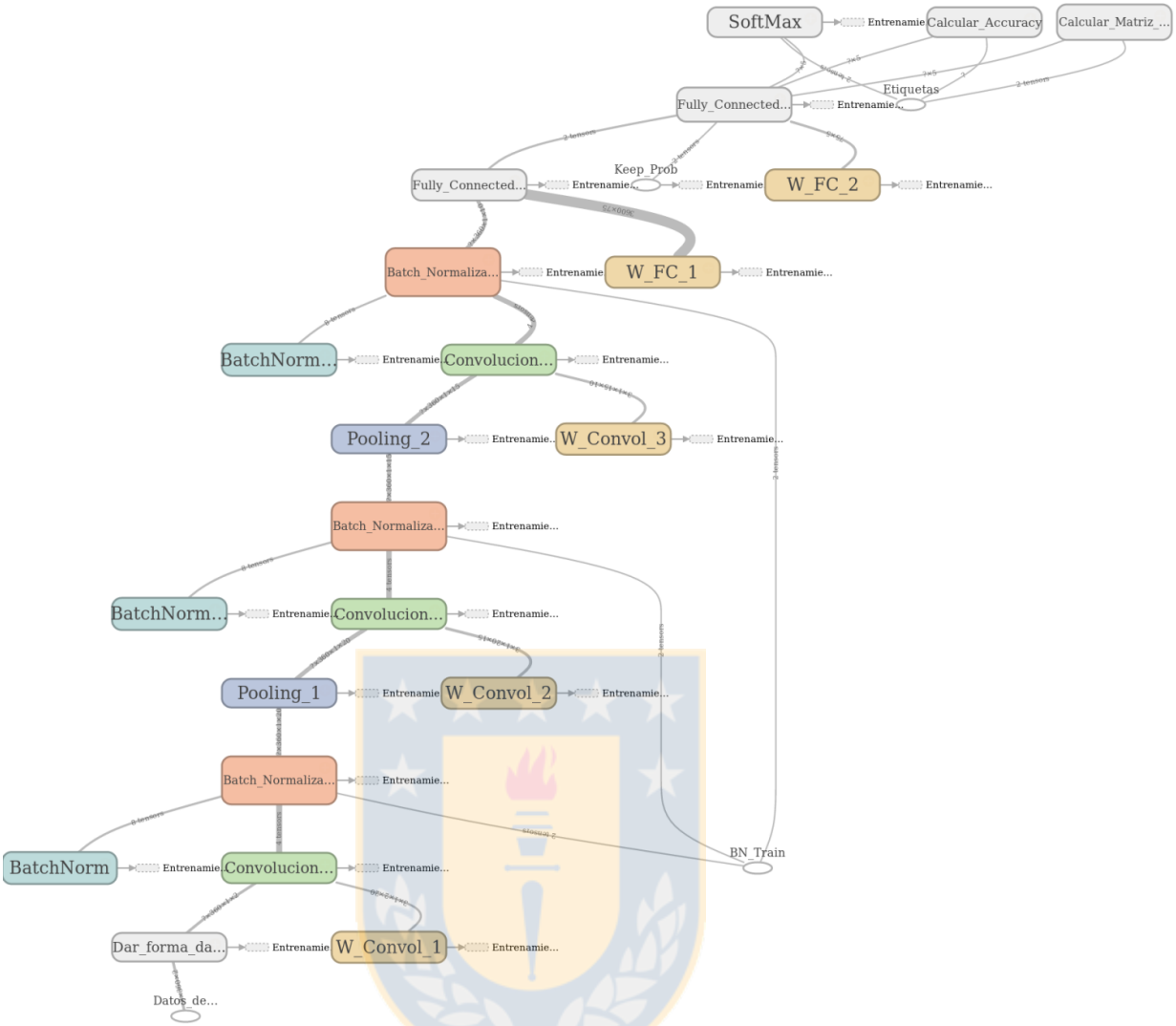

Figura 5.10 Grafo de la red CNN2F. Fuente: Elaboración propia

		Clases predichas por el modelo					Total
		N	**S**	**V**	**F**	**Q**	
Clases reales	**N**	44.047	54	129	1	0	44.231
	S	1.827	6	4	0	0	1.837
	V	730	8	2.481	1	0	3.220
	F	221	0	161	6	0	388
	Q	6	0	1	0	0	7
Total		46.831	68	2776	8	0	49.683

Tabla 5.3 Matriz de confusión para red CNN2F en conjunto de prueba. Fuente: Elaboración propia

Indicador de desempeño	Clase	Resultado
Exactitud global (Acc)	Todas	0,9367
Sensibilidad (Se)	N	0,9958
	S	0,0033
	V	0,7705
	F	0,0154
	Q	0,0000
Predictividad positiva (P+)	N	0,9406
	S	0,0882
	V	0,8937
	F	0,7500
	Q	-
Tasa de falsos positivos (FPR)	N	0,5106
	S	0,0013
	V	0,0063
	F	0,0000
	Q	0,0000
N° de variables entrenables	-	272.015
Tiempo de procesamiento [s]	-	682

Tabla 5.4 Indicadores de desempeño de red CNN2F en conjunto de prueba. Fuente: Elaboración propia

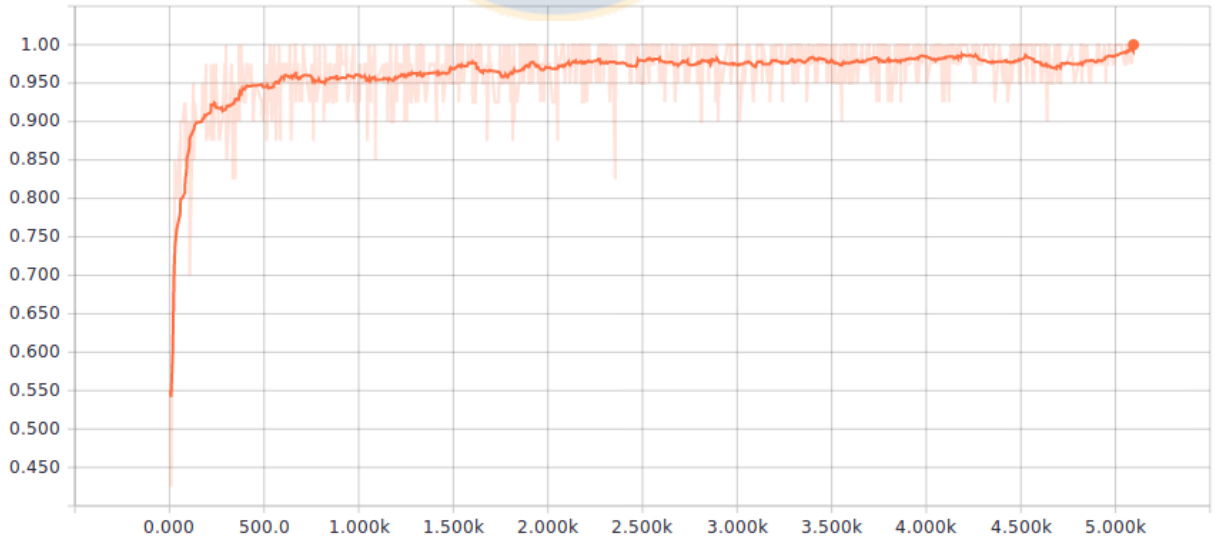

Figura 5.11 Exactitud en fase de entrenamiento de red CNN2F. Eje X: N° de iteraciones. Eje Y: Exactitud (Acc). Fuente: Elaboración propia

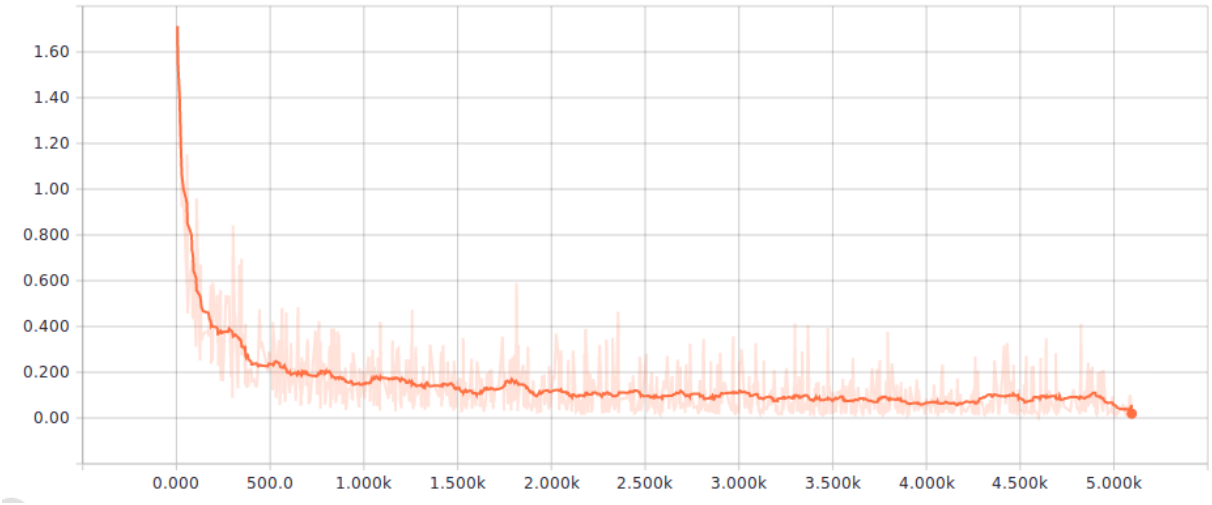

Figura 5.12 Función de pérdida en fase de entrenamiento de CNN2F. Eje X: N° de iteraciones. Eje Y: Pérdida de entropía cruzada. Fuente: Elaboración propia

5.2.3 CNN3F

El grafo de la red CNN3F se muestra en la Figura 5.13 y su descripción es la misma de la Figura 5.7, pero considerando que existen más capas; debido a su extensión, se segmentó en dos partes este grafo. La matriz de confusión y los indicadores de desempeño de la red CNN3F, se muestran en las Tablas 5.5 y 5.6, respectivamente. La evolución de la exactitud y de la función de pérdida durante el entrenamiento se muestra en las Figuras 5.14 y 5.15, cuyo gráfico se despliega en función del número de iteraciones alcanzado.

		Clases predichas por el modelo					**Total**
		N	**S**	**V**	**F**	**Q**	
Clases reales	**N**	43.967	54	200	10	0	44.231
	S	1.794	26	17	0	0	1.837
	V	296	11	2.867	46	0	3.220
	F	334	1	15	38	0	388
	Q	4	0	2	1	0	7
Total		46.395	92	3101	95	0	49.683

Tabla 5.5 Matriz de confusión para red CNN3F en conjunto de prueba. Fuente: Elaboración propia

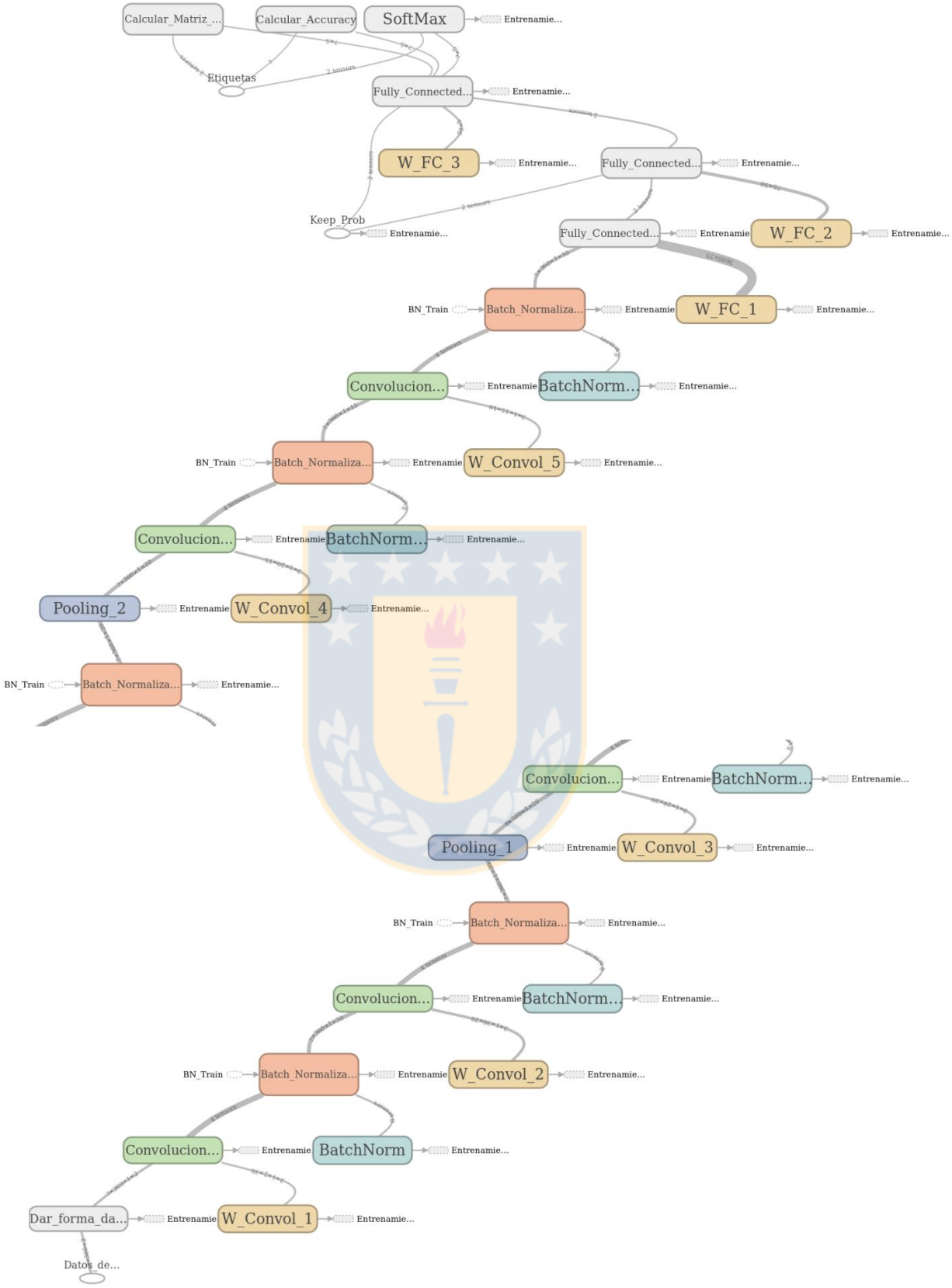

Figura 5.13 Grafo de la red CNN3F (Dividido en 2 partes por espacio). Fuente: Elaboración propia

Indicador de desempeño	Clase	Resultado
Exactitud global (Acc)	Todas	0,9439
Sensibilidad (Se)	N	0,9940
	S	0,0141
	V	0,8904
	F	0,0979
	Q	0
Predictividad positiva (P+)	N	0,9477
	S	0,2826
	V	0,9245
	F	0,4000
	Q	-
Tasa de falsos positivos (FPR)	N	0,4453
	S	0,0014
	V	0,0050
	F	0,0012
	Q	0,0000
N° de variables entrenables	-	294.690
Tiempo de procesamiento [s]	-	2.134

Tabla 5.6 Indicadores de desempeño de red CNN3F en conjunto de prueba. Fuente: Elaboración propia

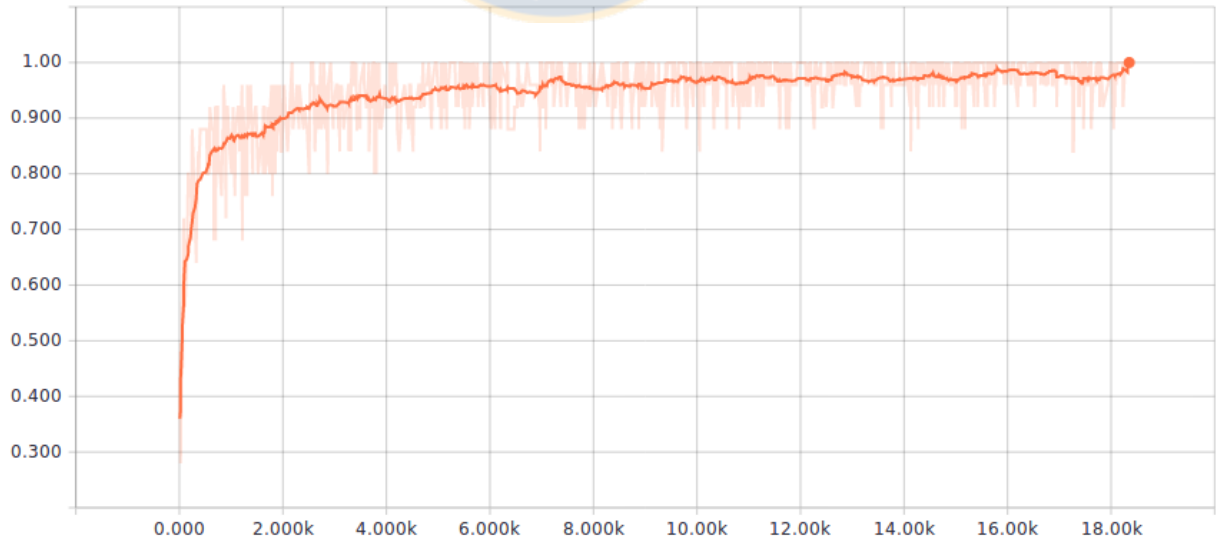

Figura 5.14 Exactitud en fase de entrenamiento de red CNN3F. Eje X: N° de iteraciones. Eje Y: Exactitud (Acc). Fuente: Elaboración propia

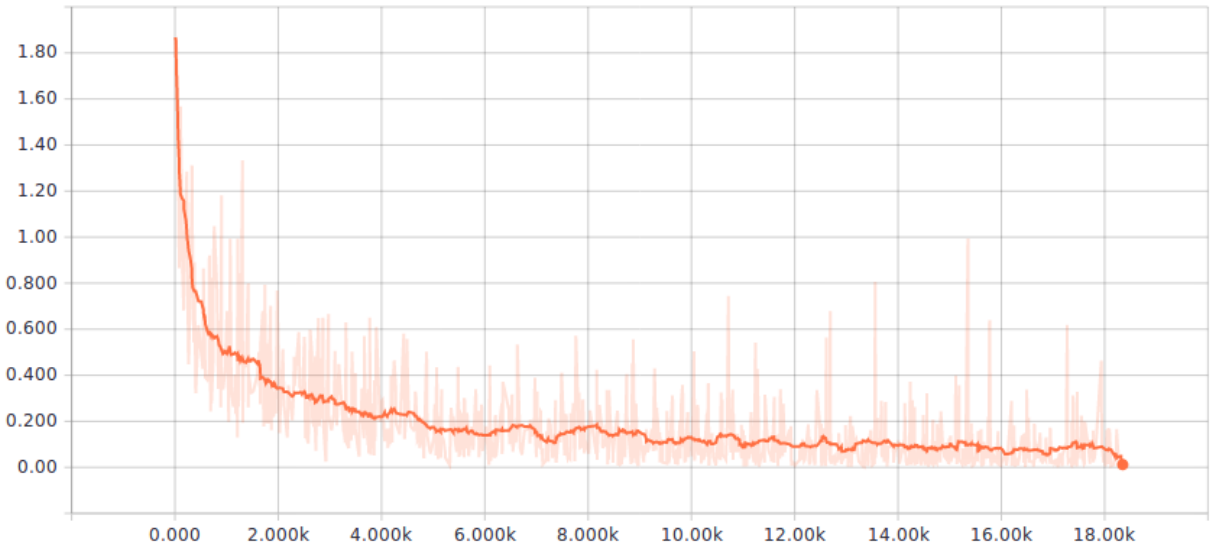

Figura 5.15 Función de pérdida en fase de entrenamiento de CNN3F. Eje X: N° de iteraciones. Eje Y: Pérdida de entropía cruzada. Fuente: Elaboración propia

5.2.4 CNN4F

El grafo de la red CNN4F se muestra en la Figura 5.16 y su descripción es la misma de la Figura 5.7, pero considerando que existen más capas; debido a su extensión, se segmentó en tres partes este grafo. La matriz de confusión y los indicadores de desempeño de la red CNN4F, se muestran en las Tablas 5.7 y 5.8, respectivamente. La evolución de la exactitud y de la función de pérdida durante el entrenamiento se muestra en las Figuras 5.17 y 5.18, cuyo gráfico se despliega en función del número de iteraciones alcanzado.

		Clases predichas por el modelo					Total
		N	S	V	F	Q	
Clases reales	N	43.972	3	247	9	0	44.231
	S	1.806	0	31	0	0	1.837
	V	465	1	2.711	43	0	3.220
	F	344	0	20	24	0	388
	Q	5	0	2	0	0	7
Total		46.592	4	3.011	76	0	49.683

Tabla 5.7 Matriz de confusión para red CNN3F en conjunto de prueba. Fuente: Elaboración propia

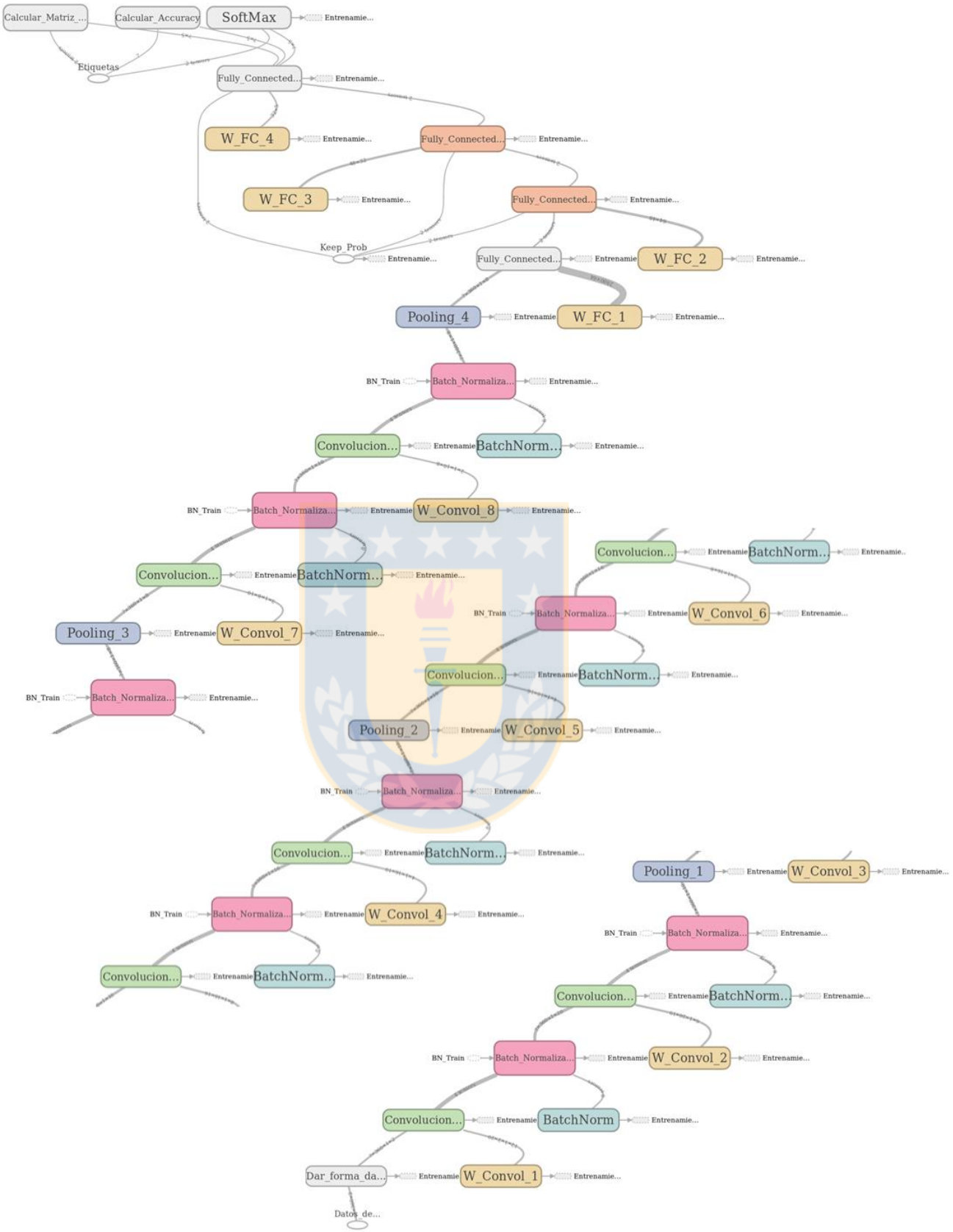

Figura 5.16 Grafo de la red CNN4F (Dividido en 3 partes por espacio). Fuente: Elaboración propia

Indicador de desempeño	Clase	Resultado
Exactitud global (Acc)	Todas	0,9401
Sensibilidad (Se)	N	0,9941
	S	0,000
	V	0,8419
	F	0,0619
	Q	0,0000
Predictividad positiva (P+)	N	0,9438
	S	0,0000
	V	0,9004
	F	0,3158
	Q	-
Tasa de falsos positivos (FPR)	N	0,4806
	S	0,0000
	V	0,0065
	F	0,0011
	Q	0,0000
N° de variables entrenables	-	115.971
Tiempo de procesamiento [s]	-	6.899

Tabla 5.8 Indicadores de desempeño de red CNN4F en conjunto de prueba. Fuente: Elaboración propia

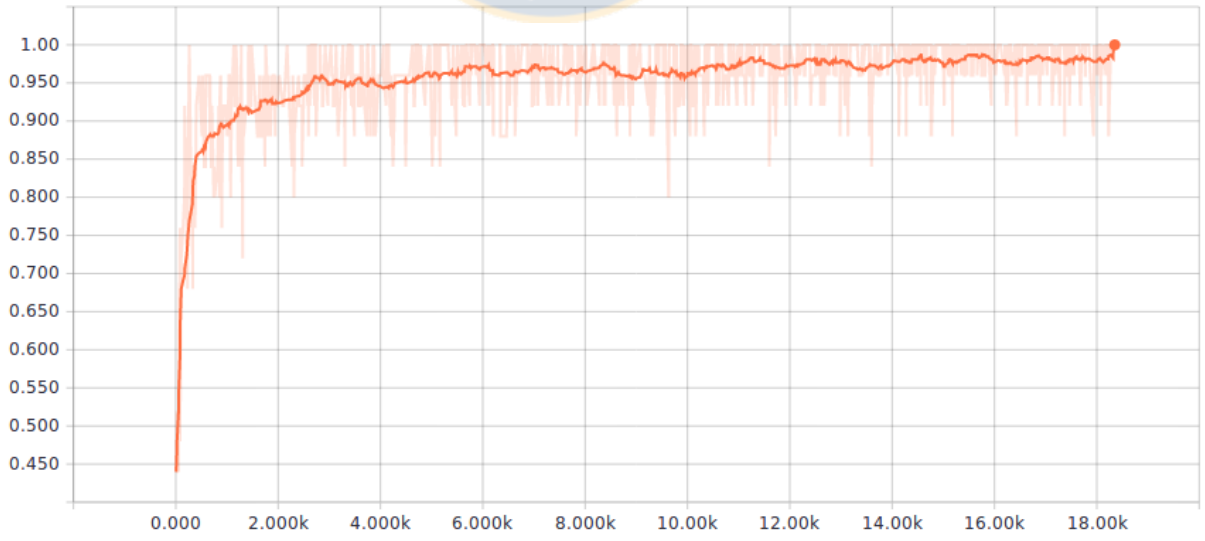

Figura 5.17 Exactitud en fase de entrenamiento de red CNN4F. Eje X: N° de iteraciones. Eje Y: Exactitud (Acc). Fuente: Elaboración propia

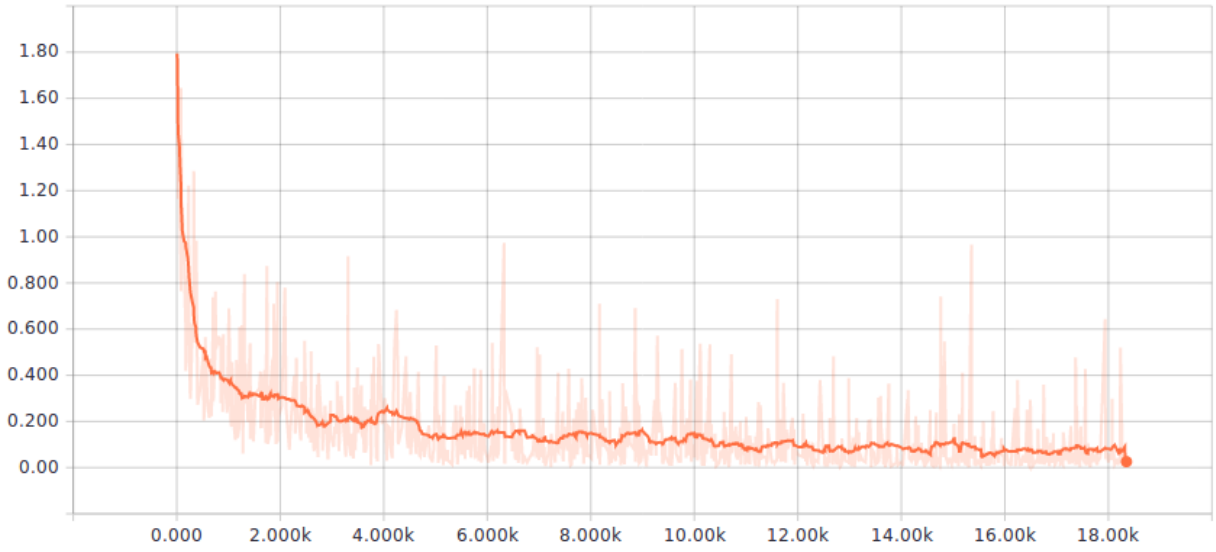

Figura 5.18 Función de pérdida en fase de entrenamiento de CNN4F. Eje X: N° de iteraciones. Eje Y: Pérdida de entropía cruzada. Fuente: Elaboración propia

5.2.5 Resumen de resultados de CNN

Con la finalidad de facilitar la comparación de los modelos implementados con los mejores de la literatura (ver Tabla 2.2), bajo el paradigma inter-paciente, se muestra un resumen de los resultados en la Tabla 5.9, donde se consideran los indicadores de desempeño usados en dichos trabajos.

Estructura	Resultados
CNN1F	Acc= 93,2% Se_N= 98,7%; $+P_N$= 94,3% Se_S= 0,4%; $+P_S$= 1,7% Se_V= 79,9%; $+P_V$= 91,7%
CNN2F	Acc= 93,7% Se_N= 99,6%; $+P_N$= 94,1% Se_S= 0,3%; $+P_S$= 8,8% Se_V= 77,1%; $+P_V$= 89,4%
CNN3F	Acc= 94,4% Se_N= 99,4%; $+P_N$= 94,8% Se_S= 1,4%; $+P_S$= 28,3% Se_V= 89,0%; $+P_V$= 92,5%
CNN4F	Acc= 94,0% Se_N= 99,4%; $+P_N$= 94,4% Se_S= 0,0%; $+P_S$= 0,0% Se_V= 84,2%; $+P_V$= 90,0%

Tabla 5.9 Resumen de resultados de CNN implementadas. Fuente: Elaboración propia

Capítulo 6. Discusión

La clasificación de latidos de un ECG es una tarea de alta complejidad. Se identifican los siguientes problemas [16]: falta de estandarización en las características de cada tipo de latido, existe variabilidad en las características de un ECG (Ej. Al variar frecuencia cardíaca), existen particularidades en los patrones de un ECG (algunos únicos por paciente), existen diferencias entre formas de latidos de un mismo tipo entre dos pacientes, hay variaciones entre los latidos de un mismo paciente, no existen reglas óptimas de clasificación para un ECG y aún no se ha encontrado el clasificador más apropiado para esta tarea.

En relación a las técnicas de aprendizaje de máquina encontradas en la literatura, cada trabajo escoge sus propia metodología de segmentación, extracción de características y clasificación; esto produce que sea muy difícil compararlos entre sí, pues no se conoce la contribución del método escogido en cada etapa al desempeño final. Este problema ya fue señalado en la revisión de Luz et al. [14].

Siguiendo con la descripción de la complejidad del problema, los autores dividen los conjuntos de entrenamiento y prueba según 3 paradigmas: intra-paciente, inter-paciente y paciente-específico. Lo que no permite una justa comparación entre las técnicas empleadas. Utilizando el esquema intra-paciente, existen latidos de un mismo registro tanto en el conjunto de entrenamiento como en el de prueba, por lo que el modelo podría estar extrayendo los patrones propios del paciente en vez de los de las clases, lo que tiene como consecuencia resultados vistosos, pero sesgados. El inter-paciente, por su parte, tiene el cuidado de considerar en un solo conjunto los latidos de un mismo registro y evitando así, que latidos de un mismo registro estén presente en los conjuntos de entrenamiento y prueba. El esquema paciente-específico es considerado como semi-automático, pues se genera un modelo general de clasificación adoptando uno de los paradigmas anteriores y luego se ajusta con latidos específicos del paciente, realizando un etiquetado manual de clases por un experto para un segmento del registro; de esta forma, se mejora notablemente los resultados de clasificación para el paciente en cuestión, pero a costa de una etapa manual, por lo que tampoco es justo compararlos con modelos automáticos. En resumen, los modelos desarrollados bajo un

paradigma no son comparables con otro y la forma correcta de proceder para la comparación justa de métodos automáticos de clasificación es adoptando el esquema inter-paciente, como el presentado en este trabajo.

De la literatura revisada (Ver Capítulo 2) con respecto a la clasificación de latidos con aprendizaje profundo, se encontró que ninguno de los modelos adoptó el esquema inter-paciente, es más, la mayoría de los trabajos eligió un enfoque paciente-específico; por lo que, pese a los buenos resultados que pudieran haber logrado, no se puede evaluar el aporte de los mismos al problema.

La base de datos recomendada por el estándar de la AAMI y la más utilizada en el entrenamiento y prueba de modelos de clasificación de latidos es la BD de arritmias del MIT-BIH. Ésta se encuentra a libre disposición en la página web de physionet [27] y para interpretar los archivos es necesario contar con la librería wfdb. Si bien los procesos y herramientas dispuestos por physionet fueron desarrollados con el fin de facilitar el acceso a las BD, resultó bastante engorroso poder manejar los registros; tanto así, que se tuvo que utilizar Ubuntu en vez de Windows como sistema operativo.

Por otro lado, la BD del MIT-BIH presenta un marcado desbalance de clases. Esto quiere decir que existe un gran número de ejemplos para la clase N, pero sólo un pequeño conjunto de las otras clases (en particular S y V). Lo anterior provoca que exista un deterioro en los resultados de clasificación, pues las clases minoritarias no están siendo bien representadas con los pocos datos disponibles. Además, la BD del MIT-BIH contiene una reducida cantidad de pacientes, lo que no permite evaluar adecuadamente la capacidad de generalización de los modelos de clasificación. Tomando en cuenta el desbalance de clases y la reducida cantidad de pacientes presentes, es que muchos autores han reportado la deficiente calidad de la BD y manifiestan la necesidad de generar nuevas BD.

En un intento de abordar el desbalance de clases, De Chazal et al. [17] sugieren una división de los conjuntos de entrenamiento y prueba que considera el esquema inter-paciente y un número similar de latidos para ambos conjuntos. Sin embargo, dado a la baja cantidad de latidos de tipo S, se incluyeron latidos de un mismo paciente en el conjunto de entrenamiento

como en el de prueba. Gran parte de los latidos de tipo S se encuentran en los registros 201 y 202 que corresponden a un mismo paciente, por lo que los modelos de clasificación podrían estar aprendiendo patrones propios del paciente en vez de buscar patrones que sirvan para generalizar la correcta clasificación de esa clase de latidos.

Pese a que en la literatura no está claro si hay o no que preprocesar, se realizó un procesamiento básico de los registros. Esta decisión se tomó debido a que las CNN en general se utilizan para el reconocimiento de patrones morfológicos [43], por lo que quitarle a la señal, los componentes conocidos de ruido que no aportan en la identificación de los latidos, podría reducir los tiempos de procesamiento durante el entrenamiento. El filtro de media móvil implementado para quitar el desplazamiento de la línea base, funciona como un filtro pasa alto con frecuencia de corte aproximada de 2[Hz], que no altera la morfología de la señal y la centra en cero. Para atenuar el ruido de la red eléctrica, se implementó un filtro digital FIR atenúa el componente de frecuencia de 60[Hz]

La segmentación de latidos se realizó con un largo fijo de 360 muestras (1[s]) con el complejo QRS en el centro. La elección de un largo fijo se debió a que la arquitectura de las CNN implementadas, en la capa de entrada, admite casos con igual número de características. Considerando que existen diferentes frecuencias cardíacas, la ventana podía enmarcar a más de 1 peak QRS o podía no abarcar un latido completo (el complejo QRS y una pequeña vecindad). Esto podría influir en la forma en que aprenden los modelos implementados, pero en la literatura en general utilizan ventanas de largo fijo para segmentar la señal de ECG [14].

Es difícil evaluar la influencia de los hiperparámetros en los resultados obtenidos, pues habría que ejecutar los algoritmos cambiando de un hiperparámetro a la vez, lo cual es poco factible dada la enorme cantidad de combinaciones posibles. En el diseño de redes profundas, la recomendación general es basarse en ciertos rangos en los que habitualmente se encuentran los hiperparámetros e ir variándolos según la experiencia del diseñador [57]. Existen métodos recientemente desarrollados para optimizar los hiperparámetros de una red neuronal profunda, pero su utilización escapa de los objetivos de este trabajo [59].

Los 4 modelos implementados presentan resultados comparables a los mejores de la literatura en cuanto a exactitud, quedando todos en el segundo puesto (Ver Tablas 2.2 y 5.9). En cuanto al detalle, los resultados de sensibilidad y predictividad positiva para las clases N y V son similares a los presentados en los trabajos utilizados en la comparación. Sin embargo, el desempeño de las CNN para la clase S está muy por debajo de los modelos de clasificación disponibles en la literatura, lo que evidencia una debilidad de las CNN para abordar el desbalance de clases de la BD usada. Si se observan las matrices de confusión presentadas (ver Tablas 5.1, 5.3, 5.5 y 5.7), se advierte claramente que los modelos clasifican la mayoría de los latidos tipo S como si fueran de tipo N, lo que indica que los modelos no aprendieron los patrones que diferencian una de otra clase. Por otro lado, la clase Q no fue incorporada en la predicción de ninguno de los modelos, pero no es de especial relevancia, pues estos latidos corresponden a casos muy raros de latidos que no son identificables y probablemente puedan ser considerados como artefactos de una señal de ECG.

Pese a que no era la que poseía más capas, la red CNN3F, presentó los mejores resultados de las CNN implementadas. De este modo, se desprende que no necesariamente una CNN más profunda obtiene los mejores resultados.

Tensorflow demostró ser una herramienta útil al implementar las CNN de este trabajo, pues incluye una gran cantidad de funciones preimplementadas que acelera tanto la curva de aprendizaje de la librería, como el desarrollo de los algoritmos. También, incorpora útiles herramientas, como Tensorboard, que provee información detallada del proceso de entrenamiento y la arquitectura de los modelos en forma de grafos. Además, posee un notable apoyo de la comunidad de programadores en el área, pues la implementación técnicas recientes en la literatura son rápidamente incluidas en el abanico de opciones de *Tensorflow*. Sin embargo, los resultados y la eficiencia de los algoritmos ejecutados en esta librería, podrían estar siendo afectados por su instalación dentro de una máquina virtual.

Capítulo 7. Conclusiones y trabajo futuro

A continuación, se presentan las principales conclusiones obtenidas de la realización de este trabajo y, a partir de ello, se propone un lineamiento y/o sugerencias sobre los asuntos que quedan por abordar para posteriores trabajos referentes a la clasificación automática de latidos desde señales de ECG.

7.1. Conclusiones

Se logró implementar un modelo de clasificación de latidos cardíacos desde señales de ECG con estructuras de aprendizaje profundo, en específico redes neuronales convolucionales (CNN), que demostraron resultados comparables a los mejores trabajos presentes en la literatura especializada del tema en cuanto a exactitud. Sin embargo, los modelos implementados demostraron ser sensibles al desbalance de clases de la base de datos utilizada, en particular con la clase S.

La clasificación de latidos cardíacos desde registros de ECG es un problema complejo, con muchas aristas y que, si bien se han desarrollado varios trabajos que abordar el tema, no es tan evidente cuál de ellos logra mejores resultados, pues hay discrepancias en la comparación y sólo son justamente comparables aquellos que adoptan un mismo paradigma (intra-paciente, inter-paciente o paciente-específico).

La base de datos de uso estándar en la tarea abordada, la BD de arritmias del MIT-BIH, presenta un claro desbalance de clases, que afecta los resultados de los modelos de clasificación, pues las clases minoritarias no son correctamente representadas. Además, son pocos los pacientes que conforman la base de datos, lo que no permite evaluar de mejor forma la capacidad de generalización de los modelos.

La elección de una ventana de largo fijo en la etapa de segmentación de latidos podría estar afectando negativamente al entrenamiento de los modelos, pues se pueden presentar más de un complejo QRS o puede que no se abarque la totalidad de un latido.

La arquitectura que entregó mejores resultados es la CNN3F, pese a no ser la más profunda, lo que indica que la profundidad de una red no necesariamente incide en un mejor desempeño de clasificación.

La librería *Tensorflow* agilizó la implementación de las estructuras de CNN y su entrenamiento, con una curva de aprendizaje bastante rápida y herramientas muy útiles, por lo que se recomienda su uso en la implementación de redes de aprendizaje profundo.

7.2. Trabajo futuro

Por las deficiencias mencionadas, es necesario enfocar esfuerzos en el desarrollo de nuevas y mejores bases de datos para la clasificación de latidos, que posean clases balanceadas y que dispongan de registros pertenecientes a un gran número de pacientes.

Queda pendiente, para la clasificación de latidos, probar otras estructuras de aprendizaje profundo, como redes neuronales recurrentes (útiles en datos secuenciales), redes de creencia profunda (extracción automática de características) u otro tipo de estructura que sea capaz de manejar características temporales y morfológicas.

Se podría implementar y evaluar el comportamiento de redes neuronales convolucionales capaces de manejar entradas de largo variable, pues se podría utilizar un conjunto de datos donde la ventana de cada latido se adaptara para efectivamente considerar las muestras pertenecientes a un solo latido

Sería de gran importancia implementar métodos para compensar el desbalance de clases, que se presenta como principal problema para mejorar los resultados, en particular para la clase S. Así como también, se podría incorporar un procedimiento de optimización para la elección de los hiperparámetros.

Tensorflow permite la implementación de los algoritmos en unidades de procesamiento gráfico (GPU), lo que reduciría considerablemente los tiempos de procesamientos y, por sus capacidades, permitiría aumentar la capacidad de representaciónde las CNN implementadas.

BIBLIOGRAFÍA

[1] Mendis, S., "Global status report on noncommunicable diseases 2014", World Health Organization, 2014.

[2] Bloom, D.E., Cafiero, E.T., Jané-Llopis, E., y Abrahams-Gessel, S., "The Global Economic Burden of Non-communicable Diseases", World Economics Furum, Génova, 2011.

[3] S. Mendis, P. Puska, B. Norrving, y others, *Global atlas on cardiovascular disease prevention and control*. World Health Organization, 2011.

[4] E. V. Ruiz de Castroviejo *et al.*, "Análisis de la frecuencia de las arritmias cardíacas y de los trastornos de conducción desde una perspectiva asistencial", *Revista Española de Cardiología*, vol. 58, nº 6, pp. 657–665, jun. 2005.

[5] C. W. Hamm y S. Willems, *El electrocardiograma: su interpretación práctica*. Ed. Médica Panamericana, 2010.

[6] L. Gaztañaga, F. E. Marchlinski, y B. P. Betensky, "Mecanismos de las arritmias cardiacas", *Revista Española de Cardiología*, vol. 65, nº 2, pp. 174–185, feb. 2012.

[7] J. L. Palma, A. Arribas, J. R. González, E. Marín, y E. Simarro, "Guías de práctica clínica de la Sociedad Española de Cardiología en la monitorización ambulatoria del electrocardiograma y presión arterial", *Revista Española de Cardiología*, vol. 53, nº 1, pp. 91–109, 2000.

[8] B. Surawicz y T. Knilans, Eds., "Normal Electrocardiogram: Origin and Description", en *Chou's Electrocardiography in Clinical Practice*, 6ta ed., Philadelphia: W.B. Saunders, 2008, pp. 1–28.

[9] D. T. Huang y T. Prinzi, Eds., "Cardiac Conduction and Bradycardia", en *Clinical Cardiac Electrophysiology in Clinical Practice*, London: Springer London, 2015.

[10] T. A. Bass, "Interventional Cardiology US Workforce Current Challenges", *Circ Cardiovasc Interv*, vol. 7, nº 6, pp. 733–735, dic. 2014.

[11] S. Aneja *et al.*, "US cardiologist workforce from 1995 to 2007: modest growth, lasting geographic maldistribution, especially in rural areas", *Health Aff (Millwood)*, vol. 30, nº 12, pp. 2301–2309, dic. 2011.

[12] G. D. Clifford, F. Azuaje, y P. McSharry, "ECG statistics, noise, artifacts, and missing data", en *Advanced Methods and Tools for ECG Data Analysis*, 6ta ed., 2006, pp. 55–99.

[13] M. Guillou, J. Carabantes C, y V. Bustos F, "Disponibilidad de médicos y especialistas en Chile", *Revista médica de Chile*, vol. 139, nº 5, pp. 559–570, may 2011.

[14] E. J. da S. Luz, W. R. Schwartz, G. Cámara-Chávez, y D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey", *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 144–164, Abril 2016.

[15] M. Velic, I. Padavic, y S. Car, "Computer aided ECG analysis; State of the art and upcoming challenges", en *2013 IEEE EUROCON*, 2013, pp. 1778–1784.

[16] S. H. Jambukia, V. K. Dabhi, y H. B. Prajapati, "Classification of ECG signals using machine learning techniques: A survey", en *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*, 2015, pp. 714–721.

[17] P. De Chazal, M. O'Dwyer, y R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features", *IEEE Transactions on Biomedical Engineering*, vol. 51, nº 7, pp. 1196–1206, 2004.

[18] O. Faust y E. Y. K. Ng, "Computer aided diagnosis for cardiovascular diseases based on ecg signals: a survey", *J. Mech. Med. Biol.*, vol. 16, nº 1, p. 164, feb. 2016.

[19] Z. Zhang, J. Dong, X. Luo, K.-S. Choi, y X. Wu, "Heartbeat classification using disease-specific feature selection", *Computers in Biology and Medicine*, vol. 46, pp. 79–89, mar. 2014.
[20] F. A. Elhaj, N. Salim, A. R. Harris, T. T. Swee, y T. Ahmed, "Arrhythmia recognition and classification using combined linear and nonlinear features of ECG signals", *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 52–63, abr. 2016.
[21] Y. Yan, X. Qin, Y. Wu, N. Zhang, J. Fan, y L. Wang, "A restricted Boltzmann machine based two-lead electrocardiography classification", en *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2015, pp. 1–9.
[22] M. Huanhuan y Z. Yue, "Classification of Electrocardiogram Signals with Deep Belief Networks", en *2014 IEEE 17th International Conference on Computational Science and Engineering (CSE)*, 2014, pp. 7–12.
[23] S. Kiranyaz, T. Ince, y M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks", *IEEE Transactions on Biomedical Engineering*, vol. 63, n° 3, pp. 664–675, mar. 2016.
[24] M. M. A. Rahhal, Y. Bazi, H. AlHichri, N. Alajlan, F. Melgani, y R. R. Yager, "Deep Learning Approach for Active Classification of Electrocardiogram Signals", *Inf. Sci.*, vol. 345, n° C, pp. 340–354, jun. 2016.
[25] E. D. Übeyli, "Combining recurrent neural networks with eigenvector methods for classification of ECG beats", *Digital Signal Processing*, vol. 19, n° 2, pp. 320–329, mar. 2009.
[26] Association for the advancement of Medical Instrumentation, "Association for the Advancement of Medical Instrumentation : ANSI/AAMI EC57/Ed.3, Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms". [En línea]. Disponible en: https://standards.aami.org/kws/public/projects/project/details?project_id=30. [Accedido: 10-jul-2016].
[27] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet", *Circulation*, vol. 101, n° 23, pp. e215–e220, jun. 2000.
[28] G. B. Moody y R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database", *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, n° 3, pp. 45–50, may 2001.
[29] A. Taddei *et al.*, "The European ST-T database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography", *European heart journal*, vol. 13, n° 9, pp. 1164–1172, 1992.
[30] "American Heart Association ECG Database DVD", *ECRI Institute*. [En línea]. Disponible en: https://www.ecri.org/components/Pages/AHA_ECG_DVD.aspx. [Accedido: 10-jul-2016].
[31] R. E. Hermes, D. B. Geselowitz, y G. C. Oliver, "Development, distribution, and use of the American Heart Association database for ventricular arrhythmia detector evaluation", *Computers in cardiology*, pp. 263–266, 1980.
[32] F. M. Nolle, F. K. Badura, J. M. Catlett, R. W. Bowser, y M. H. Sketch, "CREI-GARD, a new concept in computerized arrhythmia monitoring systems", *Computers in Cardiology*, vol. 13, pp. 515–518, 1986.
[33] G. B. Moody, W. Muldrow, y R. G. Mark, "A noise stress test for arrhythmia detectors", *Computers in cardiology*, vol. 11, n° 3, pp. 381–384, 1984.
[34] M. A. Escalona-Morán, M. C. Soriano, I. Fischer, y C. R. Mirasso, "Electrocardiogram classification using reservoir computing with logistic regression", *IEEE journal of biomedical and health informatics*, vol. 19, n° 3, pp. 892–898, 2015.

[35] C.-C. Lin y C.-M. Yang, “Heartbeat classification using normalized RR intervals and morphological features”, *Mathematical Problems in Engineering*, vol. 2014, 2014.
[36] M. Llamedo y J. P. Martínez, “Heartbeat classification using feature selection driven by database generalization criteria”, *IEEE Transactions on Biomedical Engineering*, vol. 58, nº 3, pp. 616–625, 2011.
[37] Y. Bazi, N. Alajlan, H. AlHichri, y S. Malek, “Domain adaptation methods for ECG classification”, en *Computer Medical Applications (ICCMA), 2013 International Conference on*, 2013, pp. 1–4.
[38] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, y R. Tibshirani, *The Elements of Statistical Learning, chapter 14*. Springer New York, 2001.
[39] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.
[40] “Underfitting vs. Overfitting — scikit-learn 0.18.1 documentation”. [En línea]. Disponible en: http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html. [Accedido: 02-mar-2017].
[41] D. Y. Li Deng, “Deep Learning: Methods and Applications”, NOW Publishers, may 2014.
[42] Y. LeCun, Y. Bengio, y G. Hinton, “Deep learning”, *Nature*, vol. 521, nº 7553, pp. 436–444, may 2015.
[43] Y. Bengio, “Learning Deep Architectures for AI”, *Found. Trends Mach. Learn.*, vol. 2, nº 1, pp. 1–127, ene. 2009.
[44] L. Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning”, *APSIPA Transactions on Signal and Information Processing*, vol. 3, p. e2 (29 pages), 2014.
[45] S. Min, B. Lee, y S. Yoon, “Deep Learning in Bioinformatics”, *arXiv preprint arXiv:1603.06430*, 2016.
[46] B. Xu, N. Wang, T. Chen, y M. Li, “Empirical evaluation of rectified activations in convolutional network”, *arXiv preprint arXiv:1505.00853*, 2015.
[47] X. Glorot y Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks.”, en *Aistats*, 2010, vol. 9, pp. 249–256.
[48] S. Ioffe y C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, *arXiv preprint arXiv:1502.03167*, 2015.
[49] R. Collobert y S. Bengio, “Links between perceptrons, MLPs and SVMs”, en *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 23.
[50] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, y R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.”, *Journal of Machine Learning Research*, vol. 15, nº 1, pp. 1929–1958, 2014.
[51] L. Bottou, “Stochastic gradient descent tricks”, en *Neural networks: Tricks of the trade*, Springer, 2012, pp. 421–436.
[52] Y. Chauvin y D. E. Rumelhart, *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.
[53] D. Kingma y J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.
[54] I. Goodfellow, Y. Bengio, y A. Courville, *Deep learning (adaptive computation and machine learning series)*. Cambridge: The MIT Press, 2016.

[55] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning”, en *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.
[56] *Anaconda Software Distribution*. Continuum Analytics, 2016.
[57] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures”, en *Neural networks: Tricks of the trade*, Springer, 2012, pp. 437–478.
[58] E. O. Brigham *et al.*, *The fast Fourier transform and its applications*. Prentice Hall, 1988.
[59] J. Snoek *et al.*, “Scalable Bayesian Optimization Using Deep Neural Networks.”, en *ICML*, 2015, pp. 2171–2180.