



**Clasificación de series de tiempo por medio de arquitecturas híbridas de
aprendizaje profundo**

Dany Efrain Rubiano Jiménez

Profesor guía: Gonzalo Acuña Leiva

Trabajo de titulación en con-
formidad a los requisitos
para obtener el Título de Ingeniero
Civil en Informática

Santiago – Chile

2019

© Dany Efrain Rubiano Jiménez , 2019



• Algunos derechos reservados. Esta obra está bajo una Licencia Creative Commons Atribución-Chile 3.0. Sus condiciones de uso pueden ser revisadas en:
<http://creativecommons.org/licenses/by-nc-sa/3.0/cl/>.

RESUMEN

Las series de tiempo están presentes en una gran variedad de fenómenos. Se pueden encontrar desde el análisis de mercado de valores, economía, previsión de ventas, hasta la predicción del clima. El tamaño creciente de dichos datos, así como sus características de variabilidad, alta dimensionalidad, correlación de características y dependencia temporal, desafían e impulsan el desarrollo y mejora de los métodos de minería de datos en lo referente a la predicción, clasificación e indexación (Amr, 2012).

En particular, abordando la clasificación de series de tiempo (TSC), que básicamente se puede definir como el problema de predecir las etiquetas de clase predefinidas de las series de tiempo Cui et al. (2016), el presente trabajo ¹ propone la utilización de arquitecturas híbridas de aprendizaje profundo para la clasificación de series de tiempo univariadas, mediante el acoplamiento de redes neuronales convolucionales (CNN) y long short-term memory (LSTM). Para ello se implementaron diez diversas arquitecturas variando el orden y estructura de las capas recurrentes y convolucionales, bajo enfoques lineales y por raíces, buscando obtener el mejor modelo que pudiese competir con otros modelos de la literatura. Finalmente se escogieron los modelos bilstm_resnet y bilstm_FCN, los cuales tienen como fundamento las arquitecturas ResNet y FCN, sobre los que se pudo determinar un rendimiento superior al de sus pares bajo el enfoque de aprendizaje profundo, y de vanguardia en general, superando a HIVE-COTE y COTE, que presentaban el mejor rendimiento hasta ahora en tareas de TSC.

Palabras Claves: Series de tiempo, redes neuronales, aprendizaje profundo, clasificación, modelo, arquitectura híbrida, LSTM, CNN.

¹ Proyecto de investigación interno de la Universidad de Santiago de Chile

ABSTRACT

Time series are present in a wide variety of phenomena. They can be found from stock market analysis, economics, sales forecasting, to climate prediction. The increasing size of these data, as well as their characteristics of variability, high dimensionality, correlation of characteristics and temporal dependence, challenge and drive the development and improvement of data mining methods in relation to prediction, classification and indexing (Amr, 2012).

In particular, addressing the classification of time series (TSC), which can basically be defined as the problem of predicting the predefined class labels of the time series (Cui et al., 2016), this paper² proposes the use of hybrid architectures of deep learning for the classification of univariate time series, through the coupling of convolutional neural networks (CNN) and long short-term memory (LSTM).

For this, ten different architectures were implemented, varying the order and structure of the recurrent and convolutional layers, under linear and root approaches, seeking to obtain the best model that could compete with other models of literature. Finally, the bilstm_resnet and bilstm_FCN models were chosen, which are based on the ResNet and FCN architectures, on which could be determined a performance superior to that of their peers under the focus of deep learning, and avant-garde in general, beating HIVE-COTE and COTE, which presented the best performance so far in TSC tasks.

Keywords: Time series, neural networks, deep learning, clasification, model hybrid architecture, LSTM, CNN.

²Internal research project of the Universidad de Santiago de Chile

AGRADECIMIENTOS

TABLA DE CONTENIDOS

1 Introducción	2
1.1 Antecedentes y motivación	2
1.2 Descripción del problema	2
1.3 Solución propuesta	4
1.3.1 Características de la solución	4
1.3.2 Propósito de la solución	4
1.4 Objetivos y alcance del proyecto	5
1.4.1 Objetivo general	5
1.4.2 Objetivos específicos	5
1.4.3 Alcances y limitaciones	5
1.5 Metodología y herramientas utilizadas	6
1.5.1 Metodología	6
1.5.2 Herramientas de desarrollo	7
1.6 Organización del documento	8
2 Fundamentos teóricos	9
2.1 Series de tiempo	9
2.1.1 Estacionariedad	10
2.1.2 No estacionariedad	10
2.2 Clasificación de series de tiempo	11
2.2.1 Clasificación basada en distancia	11
2.2.2 Clasificación basada en características	11
2.2.3 Clasificación basada en modelos	12
2.3 Aprendizaje profundo	12
2.3.1 Aprendizaje supervisado	13
2.3.2 Aprendizaje semi-supervisado	13
2.3.3 Aprendizaje no supervisado	13
2.3.4 Aprendizaje de refuerzo	14
2.4 Redes neuronales artificiales	14
2.4.1 Redes neuronales convolucionales	15
2.4.2 Redes neuronales recurrentes	18
Redes neuronales long short-term memory	18
2.4.3 Funciones de activación	21
Función de transferencia lineal	22
Función de transferencia no lineal	22
2.4.4 Funciones objetivo	22
Funciones objetivo de regresión	23
Funciones objetivo de clasificación	23
2.4.5 Métodos de optimización	23
2.4.6 Métodos de regularización	26
2.4.7 Métodos de entrenamiento	29
Normalización por lotes	29
Validación cruzada	30
3 Estado del arte	31
3.1 Clasificación de series de tiempo	31
3.1.1 Clasificación basada en distancias	31
3.1.2 Clasificación basada en características	32
3.1.3 Clasificación basada en modelos	33
3.2 Aprendizaje profundo, Clasificación de series de tiempo y otras problemáticas	34

3.2.1 Arquitecturas híbridas de aprendizaje profundo	35
4 Métodos	37
4.1 Método	37
4.2 Conjuntos de datos	37
4.3 Modelos de aprendizaje profundo	42
4.3.1 Desarrollo de las arquitecturas	42
Arquitecturas profundas	44
Hiperparámetros	50
Técnicas utilizadas para el entrenamiento	50
Métricas de análisis de los modelos	52
4.3.2 Selección de los modelos	54
4.3.3 Comparación con otros modelos	56
Modelos considerados	56
Comparación estadística	58
5 Resultados	60
5.1 Selección del mejor modelo implementado	60
5.1.1 Test de hipótesis para la selección del mejor modelo	73
Tests post-hoc	74
5.2 Comparaciones con modelos con enfoque de aprendizaje profundo	79
5.2.1 Test de hipótesis para la comparación de los modelos de aprendizaje profundo	82
Tests post-hoc	83
5.3 Comparaciones con otros modelos de la literatura	86
5.3.1 Test de hipótesis para la comparación con modelos de la literatura	91
Tests post-hoc	91
6 Discusión	96
6.1 Selección del mejor modelo implementado	96
6.2 Comparaciones con modelos con enfoque de aprendizaje profundo	99
6.3 Comparaciones con otros modelos de la literatura	102
7 Conclusiones	106
7.1 Conjeturas finales	106
7.2 Lineamientos Futuros	108
7.3 Reflexiones finales	109
Referencias bibliográficas	110
Apéndices	115
A Resultados generales para los modelos bajo la métrica del error	115
B Resultados generales para los modelos bajo la métrica del accuracy	118

ÍNDICE DE TABLAS

Tabla 4.1	Características de los conjuntos de datos seleccionados.	43
Tabla 4.2	Componentes de las arquitecturas implementadas.	50
Tabla 4.3	Configuración de hiperparámetros para el entrenamiento de los modelos.	51
Tabla 5.1	Coeficientes Kappa promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.	66
Tabla 5.2	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo ECG (5). Entre paréntesis su desviación estándar.	68
Tabla 5.3	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo IMAGE (13). Entre paréntesis su desviación estándar.	69
Tabla 5.4	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo MOTION (9). Entre paréntesis su desviación estándar.	69
Tabla 5.5	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SENSOR (9). Entre paréntesis su desviación estándar.	70
Tabla 5.6	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SIMULATED (5). Entre paréntesis su desviación estándar.	70
Tabla 5.7	Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SPECTRO (3). Entre paréntesis su desviación estándar.	70
Tabla 5.8	Resultados promedio clasificados con todos los conjuntos de datos para cada modelo por métrica. Entre paréntesis su desviación estándar.	71
Tabla 5.9	Promedio de ranking por cantidad de clases del conjunto de datos.	72
Tabla 5.10	Promedio de ranking por cantidad de instancias del conjunto de datos.	72
Tabla 5.11	Promedio de ranking por longitud de la serie de tiempo.	73
Tabla 5.12	Diferencias críticas entre cada uno de los modelos obtenidos por el test post-hoc de Nemenyi.	74
Tabla 5.13	Corrección de Shaffer sobre los p-valores de las comparaciones entre los pares de modelos obtenidos por el test de Friedman	76
Tabla 5.14	Accuracy promedio de diferentes modelos con enfoque de aprendizaje profundo de la literatura y de los mejores modelos implementados seleccionados.	80
Tabla 5.15	Accuracy promedio de los dominios de los conjuntos de datos para la comparación de modelos con el enfoque de aprendizaje profundo.	82
Tabla 5.16	Diferencias críticas del test de Nemenyi para las comparaciones de los modelos con el enfoque de aprendizaje profundo.	83
Tabla 5.17	Corrección de Shaffer sobre los p-valores del test de Friedman para la comparación de los modelos con el enfoque de aprendizaje profundo.	84
Tabla 5.18	Accuracy promedio de diferentes modelos de la literatura y de los mejores modelos implementados seleccionados.	87
Tabla 5.19	Accuracy promedio de los dominios de los conjuntos de datos para la comparación con los modelos de la literatura.	89
Tabla 5.20	Promedio de ranking por cantidad de clases del conjunto de datos para las comparaciones con los modelos de la literatura.	90
Tabla 5.21	Promedio de ranking por cantidad de instancias del conjunto de datos para las comparaciones con los modelos de la literatura.	90
Tabla 5.22	Promedio de ranking por longitud de la serie de tiempo para las comparaciones con los modelos de la literatura.	91
Tabla 5.23	Diferencias críticas del test de Nemenyi para las comparaciones con los modelos de la literatura.	92

Tabla 5.24 Corrección de Shaffer sobre los p-valores del test de Friedman para la comparación con modelos de la literatura.	93
Tabla A.1 Error promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.	115
Tabla B.1 Accuracy promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.	118

ÍNDICE DE ILUSTRACIONES

Figura 1.1 Metodología de la investigación.	7
Figura 2.1 Promedio de desviaciones de la temperatura global en grados centígrados (1880-2009)	9
Figura 2.2 Arquitectura de una red neuronal artificial.	15
Figura 2.3 Arquitectura de una red neuronal convolucional, consistente de capas convolucionales, capas de agrupación y capas totalmente conectadas.	16
Figura 2.4 Obtención de un mapa de características a través de la convolución de la imagen mediante la aplicación de un filtro.	17
Figura 2.5 Max-Pooling con un filtro de 2x2 y un stride de 2	18
Figura 2.6 Bloque de memoria de una LSTM.	20
Figura 2.7 Arquitectura de una LSTM.	21
Figura 2.8 Curvas de aprendizaje.	27
Figura 2.9 Dropout. Izq: Red neuronal estandar con 2 capas ocultas. Der: Red neuronal una vez aplicado la regularización por dropout.	28
Figura 2.10 Early Stopping.	29
Figura 4.1 Arquitectura 1, CNN-BiLSTM.	44
Figura 4.2 Arquitectura 2, LSTM-CNN.	45
Figura 4.3 Arquitectura 3, CNN-LSTM-Separated.	46
Figura 4.4 Arquitectura 4, Multi-CNN-BiLSTM-2.	46
Figura 4.5 Arquitectura 5, Multi-CNN-BiLSTM.	47
Figura 4.6 Arquitectura 6, CNN-LSTM-Separated-2.	47
Figura 4.7 Arquitectura 7, FCN-BiLSTM.	48
Figura 4.8 Arquitectura 8, BiLSTM-FCN.	48
Figura 4.9 Arquitectura 9, Resnet-BiLSTM.	49
Figura 4.10 Arquitectura 10, BiLSTM-Resnet.	49
Figura 5.1 Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde Adiac, hasta FaceAll.	61
Figura 5.2 Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde FaceFour, hasta MedicallImages.	62
Figura 5.3 Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde MoteStrain hasta Trace.	63
Figura 5.4 Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde TwoLeadECG, hasta Yoga.	64
Figura 5.5 Test de hipótesis de Friedman con la extensión de Iman y Davemport para la métrica kappa de todos los modelos con todos los conjuntos de datos.	73
Figura 5.6 Test post-hoc de Nemenyi para la métrica kappa de todos los modelos con todos los conjuntos de datos.	74
Figura 5.7 Gráfico de diferencias críticas basado en el test de Nemenyi entre todos los modelos.	75
Figura 5.8 Gráfico de diferencias significativas entre los modelos basado en los p-valores obtenidos de la corrección de Bergman y Hommel con el test de Friedman.	77
Figura 5.9 Test de hipótesis de Friedman con la extensión de Iman y Davemport para la comparación con modelos con el enfoque de aprendizaje profundo.	82
Figura 5.10 Test post-hoc de Nemenyi para la comparación de los modelos con el enfoque de aprendizaje profundo.	83
Figura 5.11 Gráfico de diferencias críticas basado en el test de Nemenyi para la comparación de los modelos con el enfoque de aprendizaje profundo.	84

Figura 5.12 Gráfico de diferencias significativas entre los modelos con el enfoque de aprendizaje profundo basado en los p-valores obtenidos de la corrección de Shaffer con el test de Friedman.	85
Figura 5.13 Test de hipótesis de Friedman con la extensión de Iman y Davenport para la comparación con modelos de la literatura.	91
Figura 5.14 Test post-hoc de Nemenyi para la comparación con modelos de la literatura.	92
Figura 5.15 Gráfico de diferencias críticas basado en el test de Nemenyi para la comparación con modelos de la literatura.	93
Figura 5.16 Gráfico de diferencias significativas para las comparaciones con los modelos de la literatura basado en los p-valores obtenidos de la corrección de Shaffer con el test de Friedman.	94
Figura 6.1 Mapa de activaciones sobre la clase 1 del conjunto de datos GunPoint para los modelos FCN, ResNet, bilstm_FCN y bilstm_resnet.	101
Figura 6.2 Mapa de activaciones sobre la clase 2 del conjunto de datos GunPoint para los modelos FCN, ResNet, bilstm_FCN y bilstm_resnet.	103

i7

CAPÍTULO 1. INTRODUCCIÓN

1.1 ANTECEDENTES Y MOTIVACIÓN

Las series de tiempo pueden registrar cualquier tipo de fenómeno que involucre una dependencia temporal. Como características inherentes las series de tiempo presentan variabilidad, alta dimensionalidad, correlación de características y dependencia temporal. Los avances en la recopilación de datos y las tecnologías de almacenamiento de estos han llevado a la aparición de complejos conjuntos de datos donde los ejemplos no son simples puntos de datos, sino rastros de comportamientos complejos caracterizados por series de tiempo. Por lo tanto, a medida que aumenta la complejidad de los conjuntos de datos, es necesario mejorar los métodos que comprenden su análisis desde la predicción, clasificación o indexación.

Particularmente en lo que refiere a la clasificación, en el último tiempo han proliferado distintos métodos desde la estadística y la minería de datos. Los esfuerzos recientes se basan en el enfoque de aprendizaje profundo, ya que a través de este, no se necesita una extracción de características previa. Los modelos de aprendizaje profundo "diseñan" sus propias características durante el entrenamiento. Esto es altamente deseable, ya que uno no necesita tener experiencia en el dominio de donde se originaron los datos, para poder entrenar un modelo preciso.

Ejemplos de clasificación de series de tiempo pueden ser reconocimiento de voz y gestos, análisis de mercado de valores, detección de intrusos o monitoreo de plantas industriales.

Buscando mejorar la clasificación de la series de tiempo, el presente trabajo propone la utilización de una arquitectura híbrida de aprendizaje profundo.

1.2 DESCRIPCIÓN DEL PROBLEMA

Las series temporales son una secuencia de datos que se obtienen en diferentes instantes de tiempo. Tanto las actividades humanas como la naturaleza producen series de tiempo en todo momento y en todas partes (Wang et al., 2016). Es entonces que su análisis, ya sea mediante la clasificación, indexación, agrupación o predicción, dependiendo de la problemática que se aborda, cobra gran relevancia para el estudio de los fenómenos que registran, especialmente considerando que los avances en la recopilación de datos y las tecnologías de almacenamiento de estos han llevado a la aparición de complejos conjuntos de datos donde los ejemplos no son simples puntos de datos, sino rastros de comportamientos complejos caracterizados por series de tiempo.

En particular, la clasificación de series de tiempo (TSC), el problema de predecir mediante algún método las etiquetas de clase predefinidas de las series de tiempo (Cui et al., 2016), ha encontrado multitud de aplicaciones potenciales en los campos como finanzas, industria, salud, entre muchos otros. Se entiende por etiquetas de clase como la asignación de categorías

específicas a los patrones existentes en una serie de tiempo dado su dominio. Por ejemplo, en una serie que registre las diferentes variables que involucran los movimientos de las manos en el lenguaje de señas, las etiquetas de clase serían las palabras que describen los movimientos de las manos. Dado que las series de tiempo ya tienen sus etiquetas de clase ya definidas, los algoritmos de TSC se basan en técnicas de aprendizaje supervisado.

Se destaca que a diferencia de los problemas tradicionales de clasificación, los problemas de TSC se enfrentan a las características de variabilidad, alta dimensionalidad, correlación de características y dependencia temporal que intrínsecamente presentan las series de tiempo (Amr, 2012). Esto sin duda consiste en un desafío con el que la comunidad ha lidiado durante décadas, más aun cuando la multitud de métodos que han sido desarrollados, no alcanzan la precisión y la eficiencia que se registra en otros problemas de clasificación (Cui et al., 2016), por lo que siguen atrayendo gran atención.

Por un lado, desde la estadística se han propuesto diversos métodos para estudiar datos de series de tiempo, sin embargo, casi todos ellos suponen que las series de tiempo se ajustan a un modelo lineal, y que son estacionarias o pueden reducirse a una estacionaria mediante algunas transformaciones. Por otro lado, los métodos de minería de datos ofrecieron otras soluciones, abarcando diversos enfoques. Los métodos basados en la distancia trabajan directamente en series de tiempo sin procesar con algunas medidas de similitud predefinidas como la distancia euclíadiana o la deformación temporal dinámica (DTW) para realizar la clasificación. La combinación de DTW y el clasificador k-vecinos más cercanos es uno de los enfoques más eficientes en términos de precisión.

Los métodos basados en características suponen extraer un conjunto de características que pueden representar los patrones de las series de tiempo, cuantificándolas para formar un Bag-of-Words (BoW) que se les da a los clasificadores.

Los enfoques basados en conjuntos combinan diferentes clasificadores para lograr una mayor precisión.

Sin embargo, a pesar de su desarrollo, todos los enfoques mencionados requieren mucha elaboración en preprocessamiento de datos e ingeniería de características (Wang et al., 2016).

Esfuerzos recientes de las investigaciones se basan en el enfoque de aprendizaje profundo para las tareas relacionadas, dado que este no necesita funciones hechas a mano por personas, sino que puede aprender una representación jerárquica de características a partir de datos sin procesar de forma automática (Zheng et al., 2014). Algunos métodos han explotado las redes neuronales convolucionales (CNN) para la clasificación de series temporales de extremo a extremo, como el método CNN con múltiples escalas (MCNN) para la clasificación de series temporales univariadas (Wang et al., 2016). Y otros enfoques se dan en el uso de redes neuronales long short-term memory (LSTM) diseñadas originalmente para el aprendizaje de

series de tiempo supervisadas (Bakker, 2003), (Bakker, 2003).

A pesar de ello, en base a la literatura reciente, es posible resaltar que aún no se ha explotado en profundidad la posibilidad que ofrecen los métodos de aprendizaje profundo para resolver los problemas de TSC (Fawaz et al., 2019). Es por ello, que considerando que la gran cantidad de investigaciones desarrolladas reflejan el objetivo común de aquilar conocimiento para mejorar los sistemas de clasificación, es que se requiere un método de clasificación de series de tiempo que mejoren los resultados de los métodos actuales en términos de precisión y disminución del error, y que se ponga a disposición para su uso tanto por la comunidad científica, como por las organizaciones para la extracción de conocimiento y toma de decisiones.

1.3 SOLUCIÓN PROPUESTA

1.3.1 Características de la solución

Tomando en cuenta la problemática detectada, se propone el uso de métodos de aprendizaje profundo, considerando la ventaja de que las redes neuronales profundas no necesitan una extracción de características previa del problema abordado (Zheng et al., 2014), y encontrando la necesidad de aumentar las investigaciones que exploten las particularidades en los problemas de TSC con un enfoque de aprendizaje profundo. Se plantea el uso de una arquitectura que combine dos modelos de redes neuronales para la clasificación de series de tiempo, las redes convolucionales (CNN) y las redes long short-term memory (LSTM), dos enfoques ya explotados individualmente.

A partir de ello se genera la siguiente pregunta de investigación, ¿una arquitectura híbrida de aprendizaje profundo que combine ambos tipos de redes es capaz de mejorar la clasificación de series de tiempo con respecto a los métodos existentes?

1.3.2 Propósito de la solución

El presente trabajo busca contribuir al problema de la clasificación de series de tiempo a través de la implementación de una arquitectura basada en métodos de aprendizaje profundo que mejore la precisión de la clasificación en relación a los diversos métodos existentes. El propósito final es poner al alcance de distintas comunidades un modelo de clasificación de series de tiempo, junto con promover la investigación de la comunidad científica de la minería de datos en el uso de arquitecturas de aprendizaje profundo.

1.4 OBJETIVOS Y ALCANCE DEL PROYECTO

1.4.1 Objetivo general

El objetivo general del proyecto es implementar una arquitectura híbrida de aprendizaje profundo para la clasificación de series de tiempo univariadas, comparando su desempeño con los resultados que presentan los métodos existentes desde el punto de vista del error de clasificación.

1.4.2 Objetivos específicos

1. Adquirir conocimiento de los diversos métodos existentes para la clasificación de series de tiempo, así como de los métodos de aprendizaje para la construcción de la arquitectura propuesta.
2. Desarrollar e implementar diversas arquitecturas de aprendizaje profundo en cada conjunto de datos, realizando un proceso iterativo que afine el diseño e implementación de los modelos.
3. Evaluar resultados de la clasificación de series de tiempo obtenidos a través de los modelos implementados, utilizando diversas métricas de desempeño como la concordancia, la precisión y la disminución del error, a fin de seleccionar el modelo que presente un mayor rendimiento.
4. Comparar el modelo seleccionado con otros modelos presentes en la literatura en base a la métrica del error.
5. Establecer implicaciones producto del trabajo realizado y ofrecer lineamientos para trabajos futuros.

1.4.3 Alcances y limitaciones

A continuación se listan las limitaciones y alcances de la solución propuesta:

- Para la clasificación solamente se consideran series de tiempo univariadas.
- La evaluación del modelo se desarrolla en base a los conjuntos de datos disponibles del repositorio para la clasificación de series de tiempo de la Universidad de California¹.
- Para un análisis comparativo del modelo, se utilizará principalmente la investigación de modelos previos presentes en los paper de (Wang et al., 2016), (Cui et al., 2016) y (Bagnall, 2014), los cuales presentan un estudio de contraste de varios modelos para la clasificación con las series temporales del repositorio de la Universidad de California.

¹The UEA & UCR Time Series Classification Repository, www.timeseriesclassification.com

- El modelo de clasificación desarrollado se evalúa analizando el coeficiente Kappa, curva ROC, accuracy, precisión, F!, recall y el error logarítmico de entropía cruzada.

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

1.5.1 Metodología

Tratándose de una investigación, el método por defecto a considerar es el científico. Dicho método es un proceso sistemático de investigación que consta de las siguientes partes interdependientes (Ramirez, 2004). Véase la figura 1.1.

1. **Planteamiento del problema:** fase en la cual se determina la problemática a abordar y se plantea una pregunta de investigación. En este caso, la problemática asociada es en base a la clasificación de series de tiempo.
2. **Formulación de hipótesis:** se enuncian conjeturas acerca de la solución del problema y se formula la hipótesis de investigación. En este caso la hipótesis se enmarca en que las arquitecturas híbridas de aprendizaje profundo son capaces de mejorar la clasificación de series de tiempo.
3. **Levantamiento de información y construcción del modelo:** se construye el modelo a través del cual se realizan las pruebas para validar las hipótesis. Se diseñan y realizan experimentaciones para probar si la presunción propuesta es cierta o no. El modelo en este caso corresponde a la arquitectura híbrida de aprendizaje profundo y la experimentación se hace en base a los conjuntos de datos disponibles del repositorio para la clasificación de series de tiempo de la Universidad de California. En las pruebas individuales con cada conjunto de datos se realiza validación cruzada y se varían los hiper-parámetros del modelo buscando encontrar el óptimo. El modelo varía en una constante realimentación en torno a las experimentaciones hasta depurar el diseño y la implementación, buscando obtener los resultados esperados.
4. **Análisis e interpretación de datos:** se interpretan y estudian los resultados arrojados por las experimentaciones y observaciones en las pruebas con el método desarrollado, con un análisis profundo de los datos obtenidos. El análisis comparativo se realiza en torno a los modelos presentes en la bibliografía para la clasificación de series de tiempo mediante pruebas estadísticas en base a métricas de precisión y error.
5. **Comprobación de la hipótesis:** se acepta o rechaza la hipótesis propuesta y se compara lo encontrado con lo esperado.
6. **Conclusiones:** se afianza o debilita la teoría que soporta el estudio. Se proponen nuevos enfoques para un trabajo a futuro.

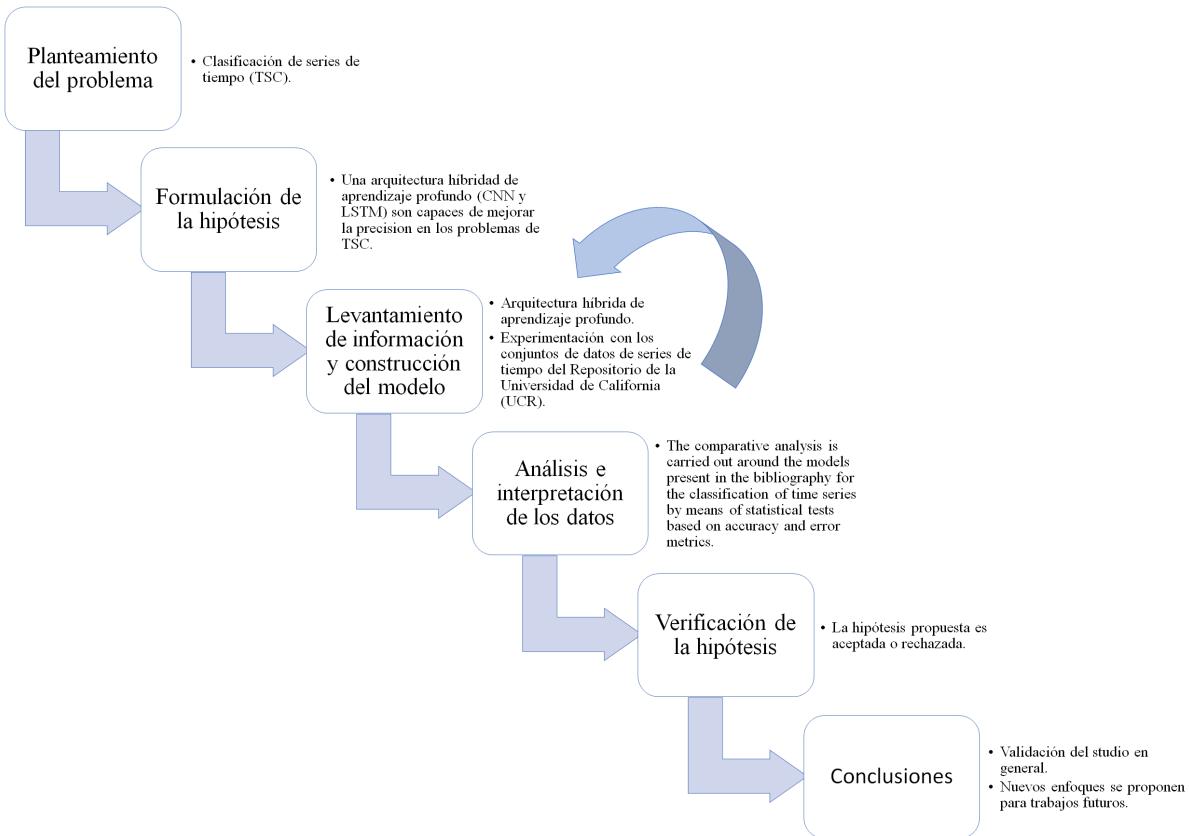


Figura 1.1: Metodología de la investigación.

Fuente: Elaboración propia.

1.5.2 Herramientas de desarrollo

En el desarrollo de este trabajo se contó con las siguientes herramientas tanto de software como hardware:

■ Software:

- Sistema operativo
- Sublime Text 3 como editor de texto para la programación².
- Lenguaje de programación Python para la implementación de la arquitectura.
- Tensorflow³, biblioteca de software libre que se utiliza para realizar cálculos numéricos mediante diagramas de flujo de datos.
- Keras⁴, biblioteca de aprendizaje profundo de Python.
- Lenguaje de programación R para la implementación de los test de hipótesis.
- R-studio, entorno de desarrollo integrado para la interacción con R.

²<https://www.sublimetext.com/3>

³<https://www.tensorflow.org/>

⁴<https://keras.io/>

- Overleaf, para la documentación del trabajo realizado⁵.
 - Google Colabs, servicio en la nube gratuito basado en Jupyter Notebooks que dispone del uso de una GPU k80.
- Hardware:
- Laptop Lenovo Z4070: Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz, 4 nucleos, 8 GB RAM y 512 GB de almacenamiento sólido.
 - Laptop HP j108-la; Intel(R) Core(TM) i7-4702MQ CPU @ 2.20GHz, 8 nucleos, 12 GB RAM y 1024 GB de almacenamiento.

1.6 ORGANIZACIÓN DEL DOCUMENTO

El presente documento se conforma por siete capítulos, que presentan todo el proceso de la investigación desarrollada, desde el establecimiento del marco teórico hasta las conclusiones finales.

El segundo capítulo presenta una explicación de los conceptos base que permiten comprender el desarrollo de la presente investigación.

En el tercer capítulo se presenta el estado del arte de las investigaciones asociadas a la problemática en general y a la solución propuesta.

En el cuarto capítulo se lleva a cabo un desarrollo de lo correspondiente al método.

Los resultados se reflejan en el quinto capítulo, donde se evalúa el modelo desarrollado con cada uno de los conjuntos de datos de series de tiempo univariadas considerados. Posteriormente el capítulo seis abarca los análisis correspondientes de los resultados con el modelo desarrollado. Finalmente en el séptimo capítulo se entregan las conclusiones respectivas al trabajo en general, verificando los objetivos propuestos inicialmente, los trabajos futuros que se podrían desarrollar y las reflexiones personales sobre el trabajo.

⁵<https://www.overleaf.com/>

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS

2.1 SERIES DE TIEMPO

Las series de tiempo son una secuencia de datos que se obtienen en diferentes instantes de tiempo a partir de la observación de un fenómeno dinámico. Tanto las actividades humanas como la naturaleza producen series de tiempo en todo momento y en todas partes (Wang et al., 2016). Ejemplo de ello se ilustra en la figura 2.1, como una serie de tiempo que registra el promedio de desviaciones de la temperatura global en grados centígrados entre los años de 1880 al 2009. Normalmente, los registros pueden ser sobre un intervalo completo, muestreados aleatoriamente en un intervalo o en puntos de tiempo fijos. Depende del tipo de muestreo el tipo de enfoque que debe ser utilizado para el análisis de datos.

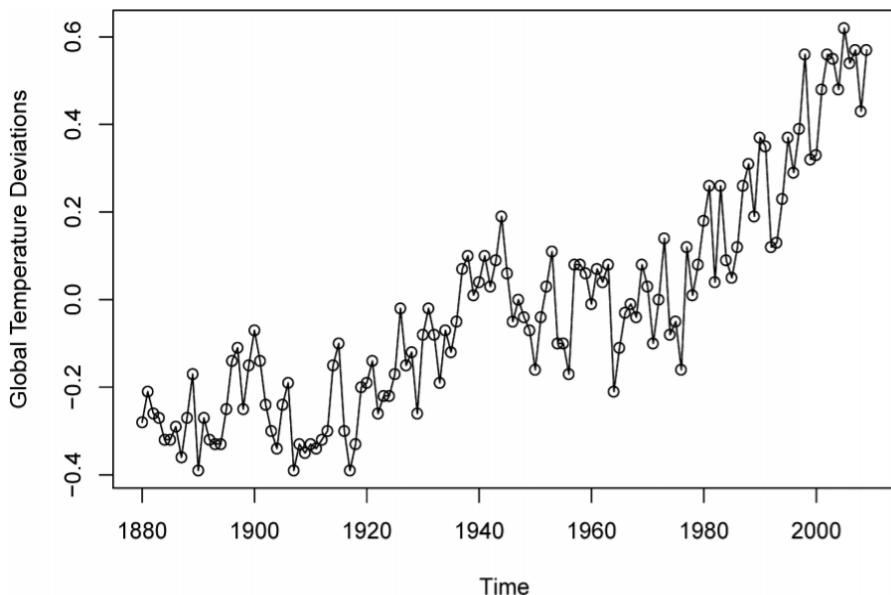


Figura 2.1: Promedio de desviaciones de la temperatura global en grados centígrados (1880-2009).

Fuente: Shumway & Stoffer (2000)

Matemáticamente, una serie de tiempo puede ser expresada en términos un conjunto de vectores x_t, t , con $x_t = x_1, x_2, x_3, \dots, x_T$ y $t = 0, 1, 2, 3, \dots, T$, donde x_t representa las observaciones en función del tiempo y t corresponde en este caso a tiempos discretos equidistantes entre si.

Una serie de tiempo que contiene registros de una sola variable es comúnmente denominada como univariada. Por otro lado, si esta considera registros de más de una variable, se denomina multivariada. A su vez, una serie de tiempo puede ser continua o discreta. En una serie de tiempo continuo, las observaciones se miden en cada instante de tiempo, mientras que

una serie de tiempo discreta contiene observaciones medidas en puntos de tiempo discretos, generalmente en intervalos equidistantes entre si, como lo pueden ser separaciones horarias, diarias, semanales, mensuales o anuales. La variable que se observa en una serie de tiempo discreta se mide como una variable continua utilizando la escala de números reales. Además, una serie temporal continua se puede transformar fácilmente en una discreta al combinar datos en un intervalo de tiempo específico (Adhikari & Agrawal, 2013).

Hay una serie de cosas que son de interés en el análisis de series de tiempo. Los más importantes de estos son (Ihaka, 2005):

Suavizado: Busca recuperar los valores de una señal suave ν_t que en conjunto con una contaminación de valores de ruido, es el resultante de la señal x_t . $x_t = \nu_t + \varepsilon_t$

Modelado: Considera el desarrollo de un modelo matemático simple que explique el patrón observado de x_1, x_2, \dots, x_T . Este modelo puede depender de parámetros desconocidos que deben ser estimados.

Pronóstico: Sobre la base de los registros x_1, x_2, \dots, x_T , busca predecir cuál será el valor de $x_T + L$ ($L \geq 1$).

Control: Pretende intervenir con el proceso que produce los valores de x_t de tal manera que los valores futuros se alteren para producir un resultado favorable.

Así mismo es de gran interés el análisis de los fenómenos dinámicos estacionarios y no estacionarios que pueden presentar las series de tiempo.

2.1.1 Estacionariedad

Una serie de tiempo es estacionaria si su comportamiento no cambia con el tiempo. Es decir, la estacionariedad implica que los valores siempre tienden a variar alrededor del mismo nivel y que su variabilidad es constante en el tiempo (Ihaka, 2005).

En términos estadísticos, se define la estacionariedad de una serie de tiempo en base a (der Vaart, 2010):

1. La serie de tiempo x_t es estrictamente estacionaria si la distribución del vector $(x_t, x_{t+1}, \dots, x_{t+h})$ es independiente de t , para cada $h \in N$.
2. La serie de tiempo x_t es estacionaria (o más precisamente de segundo orden estacionaria) si $E x_t$ y $E x_{t+h} x_t$ existen y son finitas y no dependen de t , para cada $h \in N$.

2.1.2 No estacionariedad

La no estacionariedad se refiere a la naturaleza variable en el tiempo de las distribuciones no apreciables de manera evidente. Sin embargo, muchas series de tiempo están

relacionadas de manera simple con series que son estacionarias. Dos ejemplos importantes de esto son (Ihaka, 2005):

Modelos de tendencia: la serie es la suma de una serie de tendencia determinística y una serie de ruido estacionario. Un ejemplo simple es el modelo de tendencia lineal:

$$x_t = \beta_0 + \beta_1 t + \varepsilon_t$$

Otro modelo de tendencia común asume que la serie es la suma de un efecto periódico "estacionaria" y ruido estacionario. Hay muchas otras variaciones.

Modelos integrados: la serie temporal satisface.

$$x_{t+1} - x_t = \varepsilon_{t+1}$$

donde ε_t es una serie estacionaria. Un caso particular es la caminata aleatoria, en donde los valores de ε_t son "choque" independientes que perturban el estado actual x_t en una cantidad ε_{t+1} para producir un nuevo estado x_{t+1} .

2.2 CLASIFICACIÓN DE SERIES DE TIEMPO

La clasificación de series de tiempo (TSC), puede definirse como el problema en el que se busca predecir las etiquetas de clase predefinidas de las series de tiempo. Los métodos de clasificación de series de tiempo se pueden dividir en tres categorías principales según el enfoque con el que se enfrentan, métodos basados en características, basados en modelos y basados en distancia.

2.2.1 Clasificación basada en distancia

Los métodos basados en la distancia son aquellos en los que se define una medida de similitud entre series, tales como la cuantificación mediante medidas como la distancia o correlación euclídea, para luego introducir dichas medidas distancias dentro de los métodos de clasificación basados en distancia, como el clasificador vecino más cercano k (k-NN) o las máquinas de vectores de soporte (SVM) (Abanda et al., 2018).

2.2.2 Clasificación basada en características

En los métodos de clasificación basados en características, las series de tiempo se transforman en vectores de características y luego se les aplica un clasificador convencional (Abanda et al., 2018).

Generalmente en la aplicación de este tipo de métodos, es necesario un proceso anterior de ingeniería de características, que depende de un conocimiento previo de la naturaleza del

problema que es abordado. El conjunto de características extraídas representa los patrones globales y/o locales de las series de tiempo.

2.2.3 Clasificación basada en modelos

La clasificación basada en modelos asume que todas las series de tiempo en una clase son generadas por el mismo modelo implícitamente y por lo tanto, se asigna una nueva serie con la clase del modelo que mejor se adapte. Algunos enfoques basados en modelos se forman utilizando modelos autorregresivos, modelos ocultos de Markov, o la combinación de distintos clasificadores, entre otros (Abanda et al., 2018).

2.3 APRENDIZAJE PROFUNDO

El aprendizaje profundo, es un aspecto de la inteligencia artificial (IA) que se basa en el enfoque de aprendizaje que los seres humanos utilizan para obtener ciertos tipos de conocimiento. En su forma más simple, el aprendizaje profundo puede considerarse como una forma de automatizar el análisis predictivo (Rouse, 2017).

El aprendizaje profundo considera varios niveles de composición de operaciones lineales para la función de aprendizaje, emulando la representación de manera jerárquica con múltiples niveles de abstracción con la que el cerebro humano generalmente trabaja (Bengio, 2009).

Su desarrollo se remonta desde el año 1934, en los primeros intentos por elaborar redes neuronales artificiales (RNA) (Avila et al., 2018). En 1943 McCulloch y Pitts muestran que las neuronas se pueden combinar para construir una máquina de Turing (usando ANDs, ORs y NOTs). En 1958 se constituye un hito con el primer modelo de RNA desarrollado por Rosenblatt, el perceptrón, demostrando que este convergerá si lo que intentan aprender puede ser representado. Sin embargo, años más tarde en 1969 Minsky y Papert muestran las limitaciones del perceptrón, desincentivando la investigación en RNAs por alrededor de una década (Alom et al., 2018). Posteriormente en 1986, Rumelhart, Hinton y Williams desarrollaron el algoritmo de Back-Propagation, brindando esperanza al aprendizaje de máquina y creando un hilo de aprendizaje basado en modelos estadísticos, no obstante, en su implementación con las RNA, las computadoras no tenían la capacidad computacional requerida para operar con las colecciones de datos que requerían, por lo que su éxito fue limitado. Por su parte, en la década de 1990, se propusieron una variedad de modelos de aprendizaje de máquina, como las máquinas de vectores de soporte (SVM) y la regresión logística (RL) que a la larga obtendrían un gran éxito en diversas aplicaciones (Avila et al., 2018). Fue hasta el año 2006, que en Hinton & Salakhutdinov (2006), se logró entrenar una red neuronal profunda con la capacidad de reducir la dimensionalidad de los datos, entrenando capa por capa el modelo. De esta manera, se logró obtener características que describían esencialmente la problemática abordada, facilitando el análisis de los datos. Su importancia radica en la implementación de un aprendizaje no supervisado, a través del

entrenamiento capa por capa. De aquí en adelante, en base a este trabajo se logró avanzar en las distintas limitaciones que poseían las redes neuronales artificiales, logrando al final implementar métodos con múltiples capas ocultas, dando paso al aprendizaje profundo. A partir de esto, se han logrado resultados de vanguardia en problemáticas como reconocimiento y clasificación de imágenes, procesamiento de lenguaje natural, o reconocimiento de voz (Fawaz et al., 2019).

El aprendizaje profundo puede ser categorizado en aprendizaje supervisado, semi-supervisado o parcialmente supervisado, y no supervisado. En adición, se presenta otra categoría denominado aprendizaje de refuerzo (reinforcement learning [RL] o Deep RL [DRL]) que está encasillado en la discusión bajo el alcance del enfoque de aprendizaje semi-supervisado o no supervisado (Alom et al., 2018).

2.3.1 Aprendizaje supervisado

El aprendizaje supervisado es una técnica que utiliza datos etiquetados para el proceso de entrenamiento. Es así como, el entorno tiene un conjunto de entradas y salidas correspondientes $(x_t, y_t) \sim \rho$, en el que, si para la entrada x_t , el agente inteligente predice $\hat{y}_t = f(x_t)$, este recibirá un valor de pérdida $l(y_t, \hat{y}_t)$. Luego iterativamente el agente modifica los parámetros de red para una mejor aproximación a las salidas deseadas.

Los enfoques de aprendizaje supervisado predominantes para el aprendizaje profundo, son las redes neuronales profundas, las redes neuronales convolucionales (CNN), y las redes neuronales recurrentes (RNN), que a su vez incluyen las redes con memoria a corto plazo (LSTM) y las unidades recurrentes cerradas (GRU) (Alom et al., 2018).

2.3.2 Aprendizaje semi-supervisado

El aprendizaje semi-supervisado o parcialmente supervisado, es el aprendizaje que se produce en base a conjuntos de datos parcialmente etiquetados. A menudo se le denomina aprendizaje de refuerzo. En algunos casos, las DRL y las Generative Adversarial Networks (GAN) se utilizan como técnicas de aprendizaje semi-supervisadas. Además, RNN, incluyendo LSTM y GRU, también se utilizan para el aprendizaje semi-supervisado (Alom et al., 2018).

2.3.3 Aprendizaje no supervisado

El aprendizaje no supervisado a diferencia del aprendizaje supervisado, no utiliza en el proceso de entrenamiento información alguna, como etiquetas de clase objetivo (Deng & Yu, 2013). En este caso, el agente aprende la representación interna o características importantes para descubrir relaciones o estructuras desconocidas dentro de los datos de entrada.

Casos en el que se aplica técnicas de aprendizaje no supervisado generalmente son el agrupamiento, la reducción de la dimensionalidad y las técnicas generativa (Alom et al., 2018).

Este tipo de aprendizaje genera gran interés debido a la similitud que presenta con el aprendizaje

humano y animal, los cuales generalmente se desarrollan sin supervisión. Por lo cual,

2.3.4 Aprendizaje de refuerzo

Aprendizaje de refuerzo profundo es una técnica de aprendizaje para usar en entornos desconocidos. Un ejemplo de RL es que si el entorno muestra las entradas $x_t \sim \rho$, el agente predice $\hat{y}_t = f(x_t)$, y recibe el costo $c_t \sim P(c_t|x_t, \hat{y}_t)$, donde P es una distribución de probabilidad desconocida. El entorno le hace una pregunta a un agente y le da una puntuación ruidosa como respuesta.

A veces, este enfoque también se denomina aprendizaje semi-supervisado. Hay muchas técnicas semi-supervisadas y no supervisadas que se han implementado en base a este concepto. En RL, no se tiene una función de pérdida directa, lo que dificulta el aprendizaje en comparación con los enfoques supervisados tradicionales. El aprendizaje de refuerzo no tiene acceso completo a la función que está intentando optimizar, debe consultarla a través de la interacción, además de que interactúa con un entorno basado en estados, donde la entrada x_t depende de las acciones anteriores (Alom et al., 2018).

2.4 REDES NEURONALES ARTIFICIALES

Una red neuronal artificial es un modelo que busca emular el modo en que el cerebro humano procesa la información. La red se constituye por un conjunto de neuronas interconectadas entre si, donde cada una de las neuronas están representadas por una unidad de procesamiento. Las neuronas se organizan en capas. La red cuenta normalmente con una estructura que incluye una capa de entrada, una o varias capas ocultas y una capa de salida, además de una unidad o varias unidades que representan el campo de destino, véase figura 2.2. Las unidades se conectan mediante fuerzas de conexión variables, generalmente denominadas como pesos. Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta cada neurona de la capa siguiente, así hasta que al final se envía un resultado desde la capa de salida.

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a los pesos a medida que realiza una predicción incorrecta. Esto es mediante el algoritmo de backpropagation y el método de gradiente descendiente. El proceso se repite cierto número de veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de término (IBM, 2017).

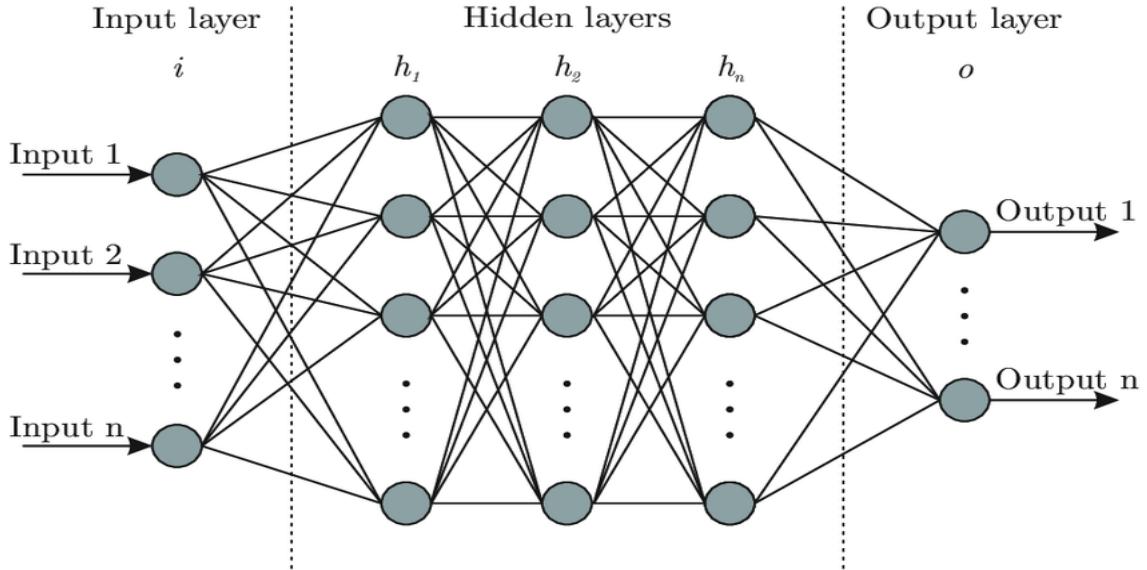


Figura 2.2: Arquitectura de una red neuronal artificial.

Fuente: Bre & Gimenez (2017)

2.4.1 Redes neuronales convolucionales

Una red neuronal convolucional (CNN por sus siglas en inglés, *convolutional neural networks*), es un modelo en el cual las neuronas corresponden a campos receptivos emulando el funcionamientos de las neuronas de la corteza visual de un cerebro biológico, específicamente están directamente inspiradas en las nociones clásicas de células simples y células complejas de la neurociencia visual, asimilándose a la jerarquía de las células en la vía ventral de la corteza visual (LeCun et al., 2015).

La característica especial de las CNN es que aprovechan la estructura espacial de las entradas (Albelwi & Mahmod, 2017), explotando la propiedad de que muchas señales naturales son jerarquías compuestas, en las cuales las características de nivel superior se obtienen al componer las de nivel inferior (LeCun et al., 2015).

En lo que se refiere a la arquitectura propiamente tal de una CNN, la red se compone de múltiples capas, véase la figura 2.2. En el principio se encuentra la fase de extracción de características compuesta de neuronas convolucionales que establecen un proceso de filtrado para la extracción de características y capas conjuntas de agrupación (pooling). A medida que se avanza en la red se disminuyen las dimensiones activando características cada vez más complejas. La capa de salida, esta compuesta por neuronas sencillas para realizar la clasificación, generalmente con una función *Softmax*.

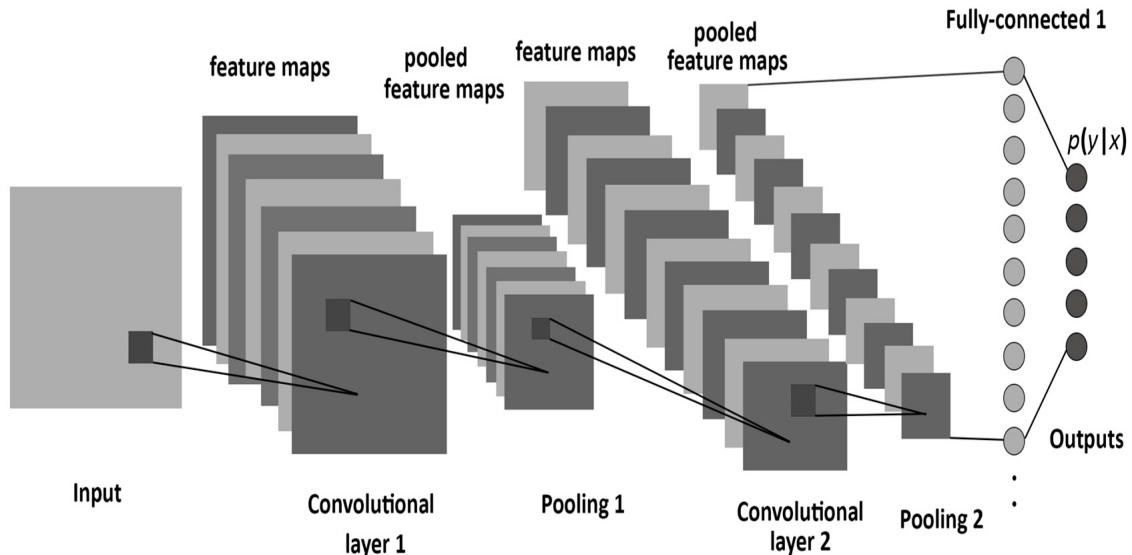


Figura 2.3: Estructura de una red neuronal convolucional, consistente de capas convolucionales, capas de agrupación y capas totalmente conectadas..

Fuente: Albelwi & Mahmod (2017)

(i) **Filtrado:**

La capa convolucional está compuesta por un conjunto de núcleos o filtros que se pueden aprender y que tienen como objetivo extraer características locales de la entrada. Cada filtro se utiliza para calcular un mapa de características, véase figura 2.3. Las unidades de los mapas de características solo pueden conectarse a una pequeña región de la entrada, denominada campo receptivo.

Un nuevo mapa de características se genera normalmente deslizando un filtro sobre la entrada y calculando la convolución con la entrada, seguido de una función de activación no lineal para introducir la no linealidad en el modelo (Albelwi & Mahmod, 2017). El filtro se desliza a través de la entrada según la cantidad que se determina en el *stride* y que generalmente se establece en un valor que resulte en un volumen de salida de número entero (Nielsen, 2016).

Todas las unidades comparten los mismos pesos (filtros) entre cada mapa de características. Las ventajas de compartir pesos, es que se utiliza un número de parámetros reducidos y que se tiene la capacidad de detectar la misma característica, independientemente de su ubicación en las entradas (Albelwi & Mahmod, 2017).

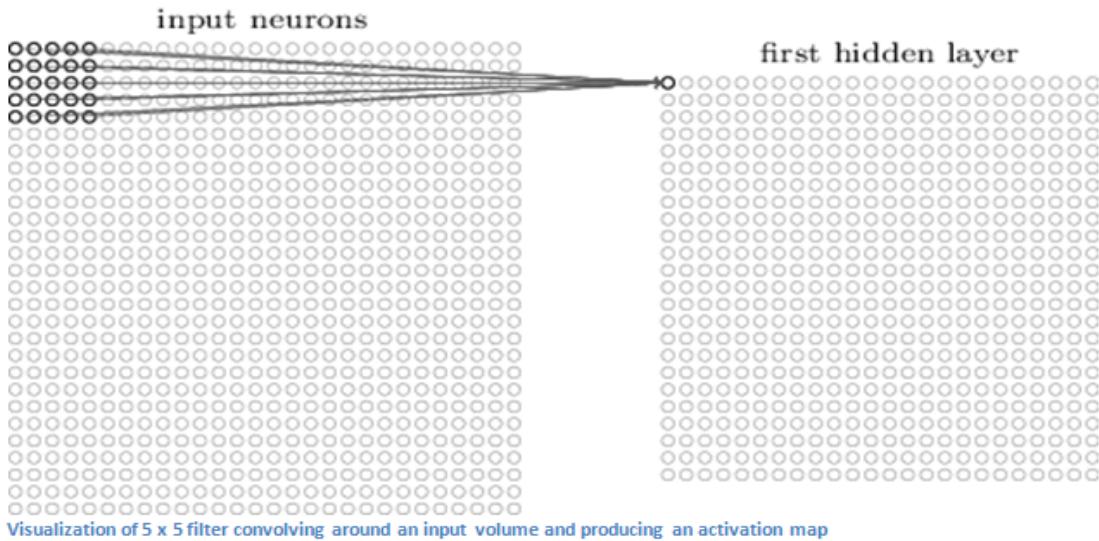


Figura 2.4: Obtención de un mapa de características a través de la convolución de la imagen mediante la aplicación de un filtro.

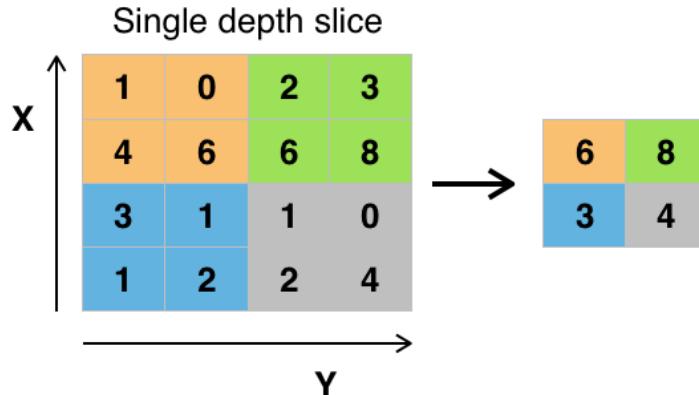
Fuente: Nielsen (2016)

(ii) Agrupación:

La agrupación, o capa de muestreo descendente, reduce la resolución de los mapas de características anteriores. El proceso divide las entradas en regiones disjuntas para producir una salida de cada región (Albelwi & Mahmod, 2017).

Esta capa sirve para dos propósitos principales. El primero es que la cantidad de parámetros o pesos se reduce en un 75 %, lo que reduce el costo de cálculo, y el segundo es que ayuda a controlar el sobreajuste (Nielsen, 2016), creando una invariancia a pequeños cambios y distorsiones (LeCun et al., 2015).

Para la agrupación se utilizan diversos métodos, entre ellos, max-pooling, average-pooling y L2-norm pooling, siendo el primero el mayormente utilizado.



Example of Maxpool with a 2x2 filter and a stride of 2

Figura 2.5: Max-Pooling con un filtro de 2x2 y un stride de 2.
Fuente: Nielsen (2016)

(iii) Clasificación:

Las capas superiores de las CNN son una o más capas completamente conectadas similares a una red neuronal de alimentación directa, que tiene como objetivo extraer las características globales de las entradas. Las unidades de estas capas están conectadas a todas las unidades ocultas en la capa anterior. La última capa es un clasificador, generalmente *Softmax*, que estima la probabilidad posterior de cada etiqueta de clase sobre determinadas clases siguiendo la estructura de un aprendizaje supervisado (Albelwi & Mahmod, 2017).

2.4.2 Redes neuronales recurrentes

Las redes neuronales recurrentes (RNN por sus siglas en inglés, *recurrent neural networks*, son un tipo de red neuronal que no solamente toman como entrada el ejemplo de la entrada actual, sino que también aquello que han percibido anteriormente. En otras palabras, la decisión de una red recurrente alcanzada en el paso del tiempo afecta la decisión en un tiempo futuro.

Debido a su circuito de retroalimentación de lazo cerrado conectado a sus decisiones pasadas, se dice que las redes recurrentes tienen memoria. La información secuencial se conserva en el estado oculto de la red, en donde se abarcan muchos pasos de tiempo a medida que avanza, de manera de afectar el procesamiento de cada nuevo ejemplo. En este proceso, se busca encontrar correlaciones entre los eventos separados por los momentos en el tiempo. Estas correlaciones se denominan "dependencias a largo plazo" (Skymind, 2017).

Redes neuronales long short-term memory

Las redes neuronales long short-term memory (LSTM), son un tipo especial de red neuronal recurrente, diseñada originalmente para el aprendizaje de series de tiempo supervi-

sadas. Se basa en un análisis de los problemas que los algoritmos recurrentes de aprendizaje de redes neuronales tienen al aprender series temporales con dependencias a largo plazo, en los que los errores propagados en el tiempo tienden a desaparecer o explotar Bakker (2003). Para ello, contienen información fuera del flujo normal de las RNN en una celda cerrada. La información puede almacenarse, escribirse o leerse desde una celda, al igual que los datos en la memoria de una computadora. La red toma decisiones sobre qué almacenar y cuándo permitir las lecturas, escrituras y eliminaciones a través de puertas que se abren y cierran. Esas puertas actúan sobre las señales que reciben, y de forma similar a los nodos de la red neuronal, bloquean o transmiten información en función de las fuerzas o pesos, que filtran con sus propios conjuntos de ponderaciones. Esos pesos, como los pesos que modulan los estados de entrada y ocultos, se ajustan a través del proceso de aprendizaje de redes recurrentes (Colah, 2015).

Específicamente, la solución que impone las LSTM a las dependencias a largo plazo, es imponer un flujo de error constante en varias unidades especializadas, denominadas carruseles de error constante (CEC) a través funciones de activación lineal que no decaen con el tiempo. Tanto el acceso de información a los CEC como su acceso desde las activaciones a las unidades de salida, son reguladas mediante compuertas de entrada y salida, las cuales reciben información de las entradas y otras unidades de la red, con la que aprenden a abrir y cerrar el acceso a los CEC, y la salida de su información almacenada en los momentos apropiados. Así mismo, incorporan compuertas de olvido, que aprenden a restablecer la activación de los CEC cuando la información que almacenan ya no es útil (Bakker, 2003). Todas las puertas son activadas mediante una función de activación, generalmente una sigmoide.

La combinación de un CEC con su entrada asociada, salida y puerta de olvido se llama celda de memoria. En la figura 2.4 se puede ver un esquema del bloque LSTM. Cuenta con tres puertas, entrada, olvido, salida, y salida cuyas ecuaciones se describen más adelante, una entrada de bloque, una sola celda (CEC), una función de activación de salida y conexiones de rendija. La salida del bloque se conecta de forma recurrente a la entrada del bloque y todas las puertas.

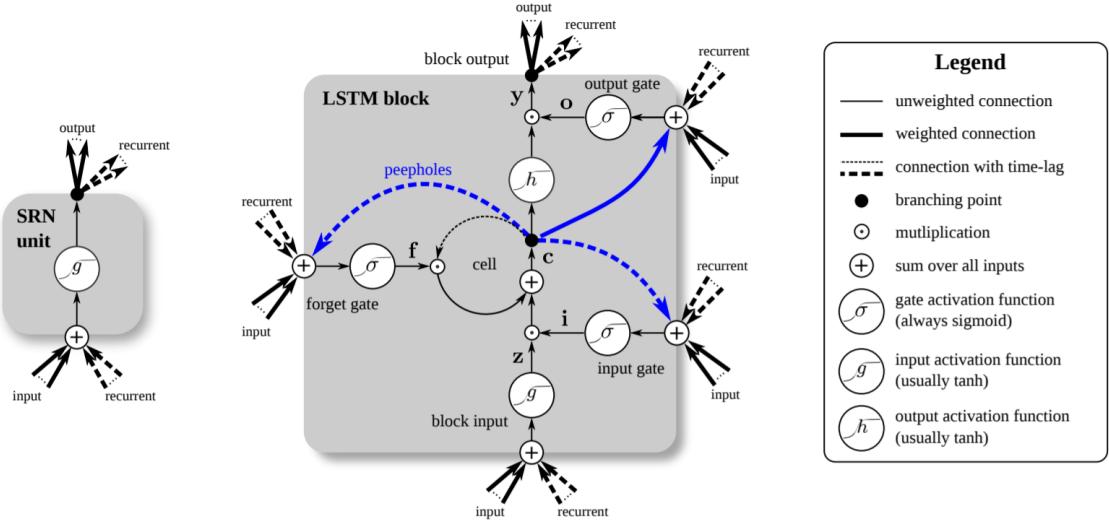


Figura 2.6: Bloque de memoria de una LSTM.
Fuente: Kang (2016)

Tomando x^t como el vector de entrada en el tiempo t , N como el número de bloques LSTM y M como el número de entradas. Los pesos de una capa LSTM corresponderían a:

- Pesos de entrada: $W_z, W_i, W_f, W_o \in R^{NxM}$
- Pesos recurrentes: $R_z, R_i, R_f, R_o \in R^{NxM}$
- Pesos rendija: $p_i, p_f, p_o \in R^{NxM}$
- Pesos Bias: $b_z, b_i, b_f, b_o \in R^{NxM}$

Matemáticamente las formulas de los vectores para una capa de LSTM son:

Bloque de entrada:

$$\begin{aligned} \bar{z}^t &= W_z x^t + R_z y^{t-1} + b_z \\ z^t &= g(\bar{z}^t) \end{aligned} \tag{2.1}$$

Puerta de entrada:

$$\begin{aligned} \bar{i}^t &= W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i \\ i^t &= \sigma(\bar{i}^t) \end{aligned} \tag{2.2}$$

Puerta de olvido:

$$\begin{aligned} \bar{f}^t &= W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f \\ f^t &= \sigma(\bar{f}^t) \end{aligned} \tag{2.3}$$

Celda:

$$c^t = z^t \odot i^t + c^{t-1} \odot f^t \tag{2.4}$$

Puerta de salida:

$$\bar{o}^t = W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o \quad (2.5)$$

$$o^t = \sigma(\bar{o}^t) \quad (2.6)$$

Salida del bloque:

$$y^t = h(c^t) \odot o^t \quad (2.7)$$

La arquitectura general de una LSTM incluye una capa de entrada, capas ocultas con bloques de memoria y finalmente la correspondiente capa de salida. En la figura 2.4, es posible ver un ejemplo de una LSTM con 8 unidades de entrada, 4 unidades de salida y 2 bloques de memoria con 2 celdas de memoria cada una.

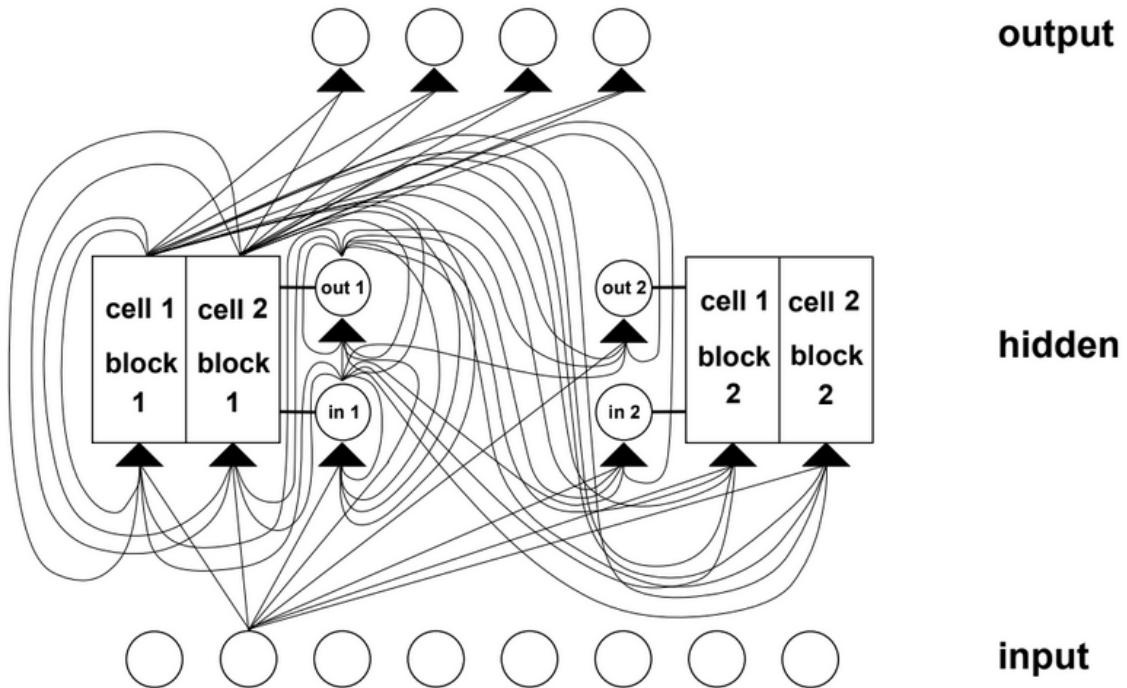


Figura 2.7: Arquitectura de una LSTM.

Fuente: Bakker (2003)

2.4.3 Funciones de activación

Las funciones de activación o de transferencia tienen como objetivo decidir si una neurona debe activarse o no. Son un proceso algorítmico que transforman el resultado de la función de propagación en la salida real de la neurona. En la función de transferencia el valor de la salida de combinación puede ser comparada con algún valor umbral para determinar la salida de la neurona. Si la suma es mayor que el valor umbral, neurona generará una señal. Si la suma es menor que el valor umbral, ninguna señal será generada (Sharma, 2017).

Existen dos tipos de funciones de transferencia según su linealidad, lineales o no

lineales.

Función de transferencia lineal

Las funciones de transferencia lineales tienen un comportamiento similar a la de una línea recta. Si en una red neuronal todas las capas son de naturaleza lineal, la función de transferencia final de la última capa no es más que una función lineal de la entrada de la primera capa (Sharma, 2017). Su forma matemática es:

$$f(x) = x$$

Básicamente, la entrada pasa a la salida sin ninguna modificación.

Función de transferencia no lineal

Las funciones de transferencia no lineales son aquellas que tienen más de un grado, por lo que su comportamiento es como el de una curva. Al utilizar una función de transferencia no lineal en una red neuronal, se pueden generar asignaciones no lineales desde las entradas a las salidas (Sharma, 2017).

Sigmoide: También conocida como función de activación logística. Toma un número de valor real y lo aplana en un rango entre 0 y 1. Matemáticamente se representa como:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Tangente hiperbólica: Similar a la sigmoide, la tangente hiperbólica (\tanh) toma un número de valor real y lo aplana en un rango entre -1 y 1, lo que permite salidas centradas en 0.

$$\tanh(x) = \frac{2}{1 + e^{2-x}} - 1$$

ReLU: La función ReLU es la unidad lineal rectificada. Es la función de activación más utilizada. Esta representada como:

$$f(x) = \max(0, x)$$

Es decir, cuando la entrada x es menor que 0, la salida es 0 ,y si x 0, la salida es x .

2.4.4 Funciones objetivo

Las funciones objetivo (loss functions) se utilizan para medir la inconsistencia entre el valor predicho (\hat{y}) y el valor real (y). Es un valor no negativo, donde la solidez del modelo aumenta a medida que disminuye el valor de la función objetivo (Parmar, 2018).

Funciones objetivo de regresión

Error cuadrático medio: Como su nombre indica, el error cuadrático medio se mide como el promedio de la diferencia cuadrada entre las predicciones y las observaciones reales. Solo le preocupa la magnitud promedio de error independientemente de su dirección.

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

Funciones objetivo de clasificación

Entropía cruzada: La función objetivo de entropía cruzada es la configuración más común para problemas de clasificación. A medida que la probabilidad predicha difiere de la etiqueta real, la función de entropía cruzada aumenta.

$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Un aspecto importante a considerar es que la función de entropía cruzada penaliza fuertemente las predicciones que son confiables pero erróneas.

2.4.5 Métodos de optimización

El descenso de gradiente es una forma de minimizar una función objetivo $J(\theta)$ parametrizada por los parámetros de un modelo $\theta \in R^d$ actualizando los parámetros en la dirección opuesta a la pendiente de la función objetivo $\nabla_{\theta} J(\theta)$ con respecto a los parámetros. La tasa de aprendizaje η determina el tamaño de los pasos que se toma para alcanzar un mínimo (local) (Agrawi, 2017).

Las variantes de los métodos de descenso de gradiente difieren entre si en términos de la cantidad de datos que utilizan para calcular el gradiente de la función objetivo. Estos son:

- Descenso de gradiente vainilla o descenso de degradado de lote, que calcula el gradiente de la función de costo de los parámetros θ para todo el conjunto de datos de entrenamiento.

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

- Descenso estocástico de gradiente (SGD, por sus siglas en inglés), realiza una actualización de parámetros para cada ejemplo $x^{(i)}$ y etiqueta $y^{(i)}$.

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

- Descenso de gradiente de mini-lotes vainilla, que combina lo mejor de los métodos anteriores, realizando una actualización para cada mini-lote de n ejemplos de entrenamiento.

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

De estos, el descenso de gradiente de mini-lotes vainilla puede llegar a una convergencia mas estable, sin embargo, se enfrenta a algunos desafíos:

- Elegir una tasa de aprendizaje adecuada puede ser difícil. Una velocidad de aprendizaje muy pequeña conduce a una convergencia lenta, mientras que una velocidad de aprendizaje muy grande puede dificultar la convergencia y hacer que la función de pérdida fluctúe alrededor del mínimo o incluso divergir.
- Como la misma tasa de aprendizaje se aplica a todas las actualizaciones de parámetros, si los datos de entrenamiento son escasos y las funciones tienen frecuencias muy diferentes, no se debería actualizarlas todas en la misma medida.
- Otro desafío en funciones de error altamente no convexas es evitar quedar atrapado en los numerosos mínimos locales subóptimos. Dauphin & Razvan Pascanu (2014) argumentan que la dificultad surge, de hecho, no desde los mínimos locales sino desde los puntos de silla, es decir, puntos donde una dimensión se inclina hacia arriba y otra hacia abajo. Estos puntos de silla generalmente están rodeados por una meseta con el mismo error, lo que hace que sea muy difícil que SGD se escape, ya que el gradiente está cerca de cero en todas las dimensiones.

Para enfrentar los desafíos mencionados anteriormente, se emplean diversos algoritmos tales como:

Momentum: es un método que ayuda a acelerar la SGD en la dirección relevante y amortigua las oscilaciones (Qian, 1999), resolviendo los problemas que tiene SGD para navegar en áreas donde la superficie se curva mucho más en una dimensión que en otra, comunes alrededor de los óptimos locales. Esto a través de la adición de una fracción γ en el vector de actualización del tiempo pasado al vector de actualización actual.

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$$

$$\theta = \theta - v_t$$

Adagrad: Algoritmo que adapta la velocidad de aprendizaje a los parámetros (Duchi et al., 2011), realizando actualizaciones más pequeñas (es decir, tasas de aprendizaje bajas)

para los parámetros asociados con las funciones que ocurren con mayor frecuencia, y actualizaciones más grandes (es decir, tasas de aprendizaje altas) para los parámetros asociados con características poco frecuentes. Por esta razón, es adecuado para trabajar con datos dispersos. Adagrad utiliza una tasa de aprendizaje diferente para cada parámetro θ_i en cada paso t .

$$g_{t,i} = \nabla_{\theta} J(\theta_{t,i})$$

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

En su regla de actualización, Adagrad modifica la tasa de aprendizaje general η en cada paso de tiempo t para cada parámetro θ_i basado en los gradientes pasados que se han calculado para θ_i .

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$G_t \in \mathbb{R}^{d \times d}$, matriz diagonal donde cada elemento diagonal i, i es la suma de los cuadrados de los gradientes con respecto a θ_i hasta el paso de tiempo t [12].

Adadelta: Adadelta es una extensión de Adagrad que busca reducir su velocidad de aprendizaje agresiva y monótonamente decreciente (Zeiler, 2012). En lugar de acumular todos los gradientes cuadrados pasados, Adadelta restringe la ventana de gradientes pasados acumulados a un tamaño fijo, entonces la suma de los gradientes se define recursivamente como un promedio en descomposición exponencial de actualizaciones de parámetros cuadrados pasados.

$$E[\Delta \theta^2]_t = \gamma E[\Delta \theta^2]_{t-1} + (1 - \gamma) \Delta \theta_t^2$$

El error cuadrático medio de las actualizaciones de parámetros es:

$$RMS[\Delta \theta]_t = \sqrt{E[\Delta \theta^2]_t + \epsilon}$$

Luego la regla de actualización del Adadelta es:

$$\Delta \theta_t = -\frac{RMS[\Delta \theta]_t - 1}{RMS[g]_t} \cdot g_t$$

$$\theta_{t+1} = \Delta \theta_t$$

RMSprop: Método de tasa de aprendizaje adaptativo que resuelve el problema de la disminución radical que tienen las tasas de aprendizaje en Adagrad.

$$E[g^2]_t = 0,9E[g^2]_{t-1} + (0,1)g_t^2$$

$$\theta_{t+i} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

Adam: La estimación del momento adaptativo (Adam) es otro método que calcula las tasas de aprendizaje adaptativo para cada parámetro (Kingma & Ba, 2015). Además de almacenar un promedio de decaimiento exponencial de gradientes cuadrados pasados v_t al igual que Adadelta y RMSprop, Adam también mantiene un promedio exponencial decadente de gradientes pasados m_t , similar al impulso. Se calcula los promedios decrecientes de los gradientes cuadrados pasados y pasados m_t y v_t respectivamente como sigue:

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

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

m_t y v_t son estimaciones del primer momento (la media) y el segundo momento (la varianza no centrada) de los gradientes respectivamente, de ahí el nombre del método. Para evitar el sesgo hacia cero, debido a la inicialización de los vectores en 0, se calculan las estimaciones del primer y segundo momento como:

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

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

Estas son usadas para actualizar los parámetros, produciendo la siguiente regla de actualización.

$$\theta_{t+i} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t$$

2.4.6 Métodos de regularización

Los modelos neuronales tienen como objetivo lograr la generalización, esto es que tengan un buen desempeño tanto con los datos a través de los que se capacitan, como en los datos nuevos. Dependiendo del aprendizaje y la generalización, se puede clasificar los modelos como underfit (infraajuste), overfit (sobreajuste) y good-fit (ajuste correcto).

Underfit: Modelo que tiene un alto sesgo y baja varianza. En otras palabras, es un modelo que presenta un desempeño deficiente con el conjunto de datos de entrenamiento y consecuentemente no tiene un buen desempeño con una muestra de nuevos datos.

Overfit: Modelo que tiene sesgo bajo y alta varianza. Es decir, es un modelo que presenta un gran desempeño con el conjunto de datos de entrenamiento, pero que no se desempeña bien en una muestra de nuevos datos.

Good-fit: Modelo que tiene un aprendizaje adecuado con el conjunto de datos de entrenamiento y generaliza bien un conjunto de datos nuevo.

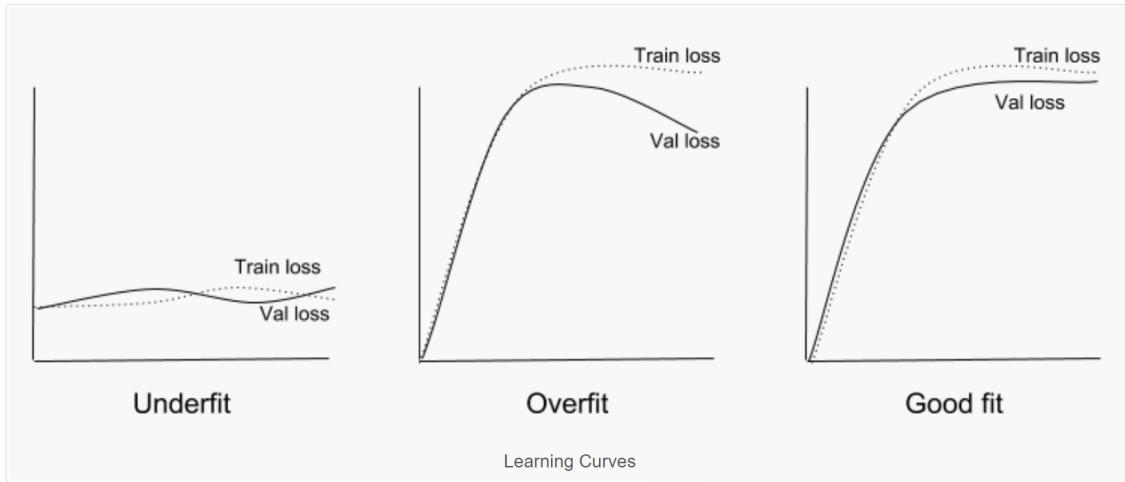


Figura 2.8: Curvas de aprendizaje.

Fuente: Brownlee (2018)

Para afrontar el infra-ajuste, basta con aumentar la capacidad del modelo, a través de la variación de su estructura, esto puede ser mediante la adición de más capas o el aumento de los nodos por capa.

En el caso del sobreajuste, se ocupa la regularización, la cual consiste básicamente en modificar el algoritmo de aprendizaje, buscando reducir el error de generalización. A continuación se detallan algunos de los métodos de regularización más comunes.

Regularización de L2 y L1: Son los tipos más comunes de regularización. Éstos actualizan la función objetivo general agregando otro término conocido como el término de regularización. Esto hace que los valores de las matrices de peso disminuyan porque asume que una red neuronal con matrices de menor peso conduce a modelos más simples (Jain, 2018). De esta manera, también se reduce el sobreajuste en gran medida.

L1 y L2 difieren entre sí por el término de regularización. *L2*:

$$\text{Cost_function} = \text{Loss} + \frac{\lambda}{2m} \sum ||w||^2$$

λ es el parámetro de regularización, cuyo valor se optimiza para obtener mejores resultados. La regularización de L2 también se conoce como caída de peso, ya que obliga a los pesos a caer hacia cero (pero no exactamente a cero).

L1:

$$\text{Cost_function} = \text{Loss} + \frac{\lambda}{2m} \sum ||w||$$

L1 penaliza el valor absoluto de los pesos, los cuales se pueden reducir casi a 0. Esto es útil a la hora de comprimir el modelo.

Dropout: Es la técnica de regularización más utilizada en el campo del aprendizaje profundo. Su funcionamiento radica en que en cada iteración durante el proceso de aprendizaje, selecciona aleatoriamente algunos nodos y los elimina junto con todas sus conexiones entrantes y salientes (Jain, 2018). De esta forma, cada iteración presenta un conjunto diferente de nodos, lo que resulta en un conjunto diferente de salidas.

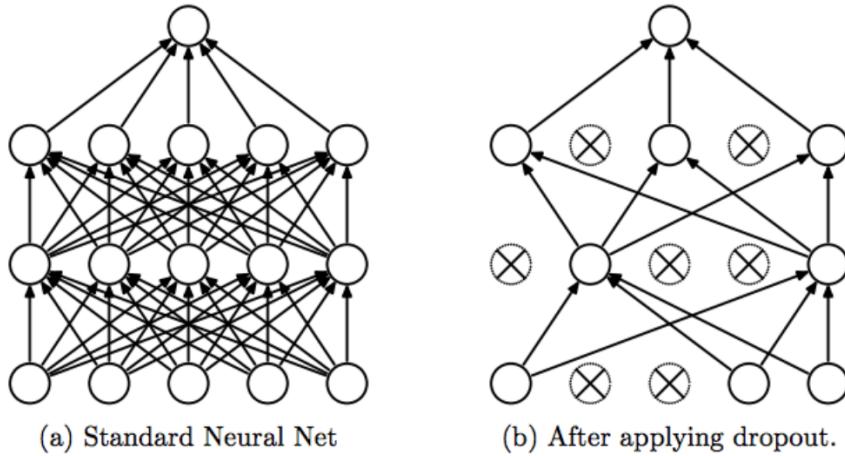


Figura 2.9: Dropout.(a) Red neuronal estandar con 2 capas ocultas. (b) Red neuronal una vez aplicado la regularización por dropout.

Fuente: Jain (2018)

La función Dropout se ajusta mediante un hiperparámetro que indica la probabilidad de elegir cuantos nodos deben eliminarse.

Early Stopping: Es un tipo de estrategia de validación cruzada, donde se reserva una muestra del conjunto de entrenamiento como el conjunto de validación. De esta forma se puede monitorear el rendimiento del modelo en cada iteración, deteniendo el aprendizaje cuando el rendimiento en el conjunto de validación una vez llegado a su tope, comienza a disminuir (Jain, 2018).

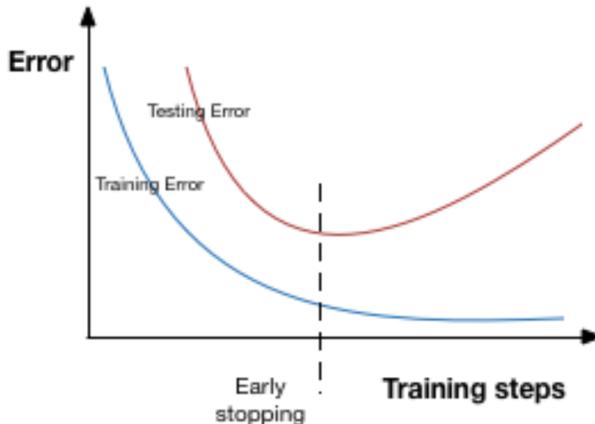


Figura 2.10: Early Stopping.
Fuente: Jain (2018)

2.4.7 Métodos de entrenamiento

Normalización por lotes

El entrenamiento en las redes neuronales profundas se complica por el hecho de que la distribución de las entradas de cada capa cambia durante el entrenamiento, a medida que cambian los parámetros de las capas anteriores. Es decir, un cambio de covarianza interna. Esto ralentiza el entrenamiento al requerir tasas de aprendizaje más bajas y una cuidadosa inicialización de parámetros, además de que hace difícil el entrenamiento de los modelos con saturaciones no lineales (Ioffe & Szegedy, 2015). La normalización por lotes es un método que ayuda a coordinar la actualización de múltiples capas en el modelo, permitiendo que cada capa de una red aprenda por sí misma un poco más independientemente de otras capas (Browniee, 2019). Para aumentar la estabilidad de una red neuronal, la normalización de lotes normaliza la salida de una capa de activación previa restando la media del lote y dividiendo por la desviación estándar del lote (Doukkali, 2017). Después del entrenamiento, la media y la desviación estándar de las entradas para la capa se pueden establecer como valores medios observados en el conjunto de datos de entrenamiento.

Entre los beneficios que permite este método se encuentra que la normalización por lotes permite utilizar una alta tasa de aprendizaje, sin que los gradientes exploten o desaparezcan, debido a que los pequeños cambios en el parámetro a una capa no se propagan a otras capas. Así mismo, la estabilidad del entrenamiento traída por la normalización de lotes puede hacer que el entrenamiento en redes profundas sea menos sensible a la elección del método de inicialización de peso (Browniee, 2019). Adicionalmente, dado que el paso de normalización ve juntos todos los ejemplos de entrenamiento en el lote, trae consigo un efecto de regularización. (Doukkali, 2017).

Validación cruzada

La validación cruzada es un procedimiento de remuestreo que se utiliza para evaluar modelos con una muestra de datos limitada. El procedimiento tiene un único parámetro llamado k que se refiere al número de grupos en que se dividirá una muestra de datos dada.

La validación cruzada se usa en el aprendizaje profundo para estimar la habilidad de un modelo en datos invisibles. Es decir, usar una muestra limitada para estimar cómo se espera que el modelo se desempeñe en general cuando se usa para hacer predicciones sobre los datos que no se usaron durante la capacitación del modelo. Esto generalmente resulta en una estimación menos sesgada o menos optimista de la habilidad del modelo que otros métodos, como una simple división de muestras de entrenamiento y prueba (Browniee, 2018).

El procedimiento general es el siguiente:

1. Mezclar el conjunto de datos al azar.
2. Dividir el conjunto de datos en k grupos
3. Para cada grupo:
 - i Tomar el grupo como un conjunto de datos de espera o prueba
 - ii Tomar los grupos restantes como un conjunto de datos de entrenamiento
 - iii Ajustar un modelo en el conjunto de entrenamiento y evaluarlo en el conjunto de prueba
 - iv Conservar el puntaje de evaluación y descartar el modelo.
4. Resumir la habilidad del modelo utilizando la muestra de puntajes de evaluación del modelo.

CAPÍTULO 3. ESTADO DEL ARTE

3.1 CLASIFICACIÓN DE SERIES DE TIEMPO

La clasificación de series de tiempo (TSC), entendida como un problema de aprendizaje supervisado en el que se busca predecir las etiquetas de clase predefinidas de las series de tiempo, ha proliferado durante décadas dentro de la comunidad de minería de datos, encontrando multitud de aplicaciones importantes como la ingeniería biomédica y la predicción clínica. Aún así, sigue siendo un desafío, puesto que no alcanza la precisión y la eficiencia de los métodos de clasificación desarrollados en otras problemáticas distintas a las series de tiempo (Cui et al., 2016). Esto, debido a que los problemas de TSC se diferencian de los tradicionales debido a sus características de variabilidad, alta dimensionalidad, correlación de características y dependencia temporal(Amr, 2012).

Tal como se desarrolla en la sección 2.2, los enfoques en los que se cimientan los modelos de clasificación de series de tiempo son basados en distancia, basados en modelos y basados en características. A continuación se presentan algunos avances en las investigaciones de este tipo de modelos basado en alguno de los enfoques mencionados.

3.1.1 Clasificación basada en distancias

Como la línea de base más antigua, la mayoría de la investigación de la clasificación de series de tiempo se ha concentrado en las distancias en el tiempo. Esto, a través de la cuantificación mediante medidas como la distancia o correlación euclíadiana. La similitud en el tiempo se caracteriza por la situación particular de que las series de cada clase son observaciones de una curva común implícita en la dimensión del tiempo. La variación en torno a esta forma común implícita es causada por el ruido en la observación, además de una posible desalineación en los ejes de tiempo que pueden causar un ligero cambio de fase. Aquellas medidas elásticas que permiten una cierta realineación pueden compensar este cambio (Bagnall, 2014). Se distinguen algunos enfoques de métodos basados en distancia, k-NN, y métodos basados en nucleos (kernels).

k-vecinos más cercanos (k-NN) es un enfoque que emplea las distancias de series de tiempo existentes dentro de los clasificadores k-NN. En particular, el clasificador 1-NN se ha utilizado principalmente en la clasificación de series de tiempo debido a su simplicidad y rendimiento competitivo. Dada una medida de distancia y una serie de tiempo, el clasificador 1-NN predice la clase de esta serie como la clase del objeto más cercano desde el conjunto de entrenamiento. Sin embargo, es muy sensible al ruido en el conjunto de entrenamiento, que es una característica común de los conjuntos de datos de series de tiempo. Este enfoque se ha aplicado ampliamente en la clasificación de series de tiempo, ya que logra, junto con la distancia

DTW (Distance Time warming), la mejor precisión alcanzada en muchos conjuntos de datos de referencia (Abanda et al., 2018). DTW con conjunto de deformación a través de validación cruzada (DTWCV) es también un punto de referencia tradicional. Los clasificadores k-NN también usan características transformadas. Bag-of-SFA-Symbols (BOSS) (Schäfer, 2015a) propone una distancia basada en los histogramas de palabras simbólicas de aproximación de Fourier. Mientras que el método BOSSVS (Schäfer, 2015b) combina el modelo BOSS con el modelo de espacio vectorial para reducir la complejidad del tiempo y mejorar el rendimiento al ensamblar los modelos con un tamaño de ventana diferente. Todos se combinan con 1-NN para la predicción final.

Por otro lado, se incluyen los métodos que no emplean las distancias de series de tiempo existentes para obtener una nueva representación de las series, sino que, las utilizan para obtener un núcleo para series de tiempo. Para la clasificación de series de tiempo, la mayor parte del trabajo se centra en emplear los núcleos de distancia propuestos por Haasdonk & Bahlmann (2004), y generalmente proponen reemplazar la distancia euclídea en las funciones tradicionales del kernel, como un kernel gaussiano, por la medida de distancia específica del problema (Abanda et al., 2018).

3.1.2 Clasificación basada en características

Los tipos de métodos basados en este enfoque, suponen extraer un conjunto de características que pueden representar los patrones de series de tiempo globales y/o locales Wang et al. (2016), luego se les aplica cualquier clasificador convencional basado en características tal como SVM o regresión logística, por decir algunos ejemplos, para generar los resultados de clasificación.

Los enfoques basados en características difieren principalmente en las características extraídas. En Nanopoulos et al. (2001) se utilizan características como la desviación estándar, la asimetría y la curtosis de la serie temporal y sus incrementos sucesivos para representar y clasificar los patrones del gráfico de control; en Mörchen (2003) se utiliza un enfoque de características derivadas de las transformadas de wavelet y Fourier de un rango de series temporales conjuntos de datos para clasificarlos; en Wang et al. (2007) se introduce un conjunto de trece características que contiene medidas de tendencia, estacionalidad, periodicidad, correlación en serie, sesgo, curtosis, caos, no linealidad y auto-similitud para representar series de tiempo; y en Deng et al. (2013a) se utilizan medidas de la media, la propagación y la tendencia en intervalos de series de tiempo locales para clasificar diferentes tipos de series de tiempo.

Otros métodos para la extracción de características incluyen métodos espectrales como la transformada discreta de Fourier (DFT) (Faloutsos et al., 1994); o la transformada discreta wavelet (DWT) (Popivanov & Miller, 2002), donde se consideran las características del dominio de la frecuencia; o la descomposición del valor singular (SVD) (Korn et al., 1997), donde se realiza el análisis de valores propios para encontrar un conjunto óptimo de características. Un enfoque

de referencia reciente es Bag-of-Temporal-SIFT-Words (TSBF) (Bailly et al., 2016) el cual extrae las funciones de intervalo con diferentes escalas para formar una instancia y formar una bolsa con cada serie de tiempo. Luego, construye un libro de códigos supervisado con random forest para la clasificación.

3.1.3 Clasificación basada en modelos

Existe una tendencia creciente a reunir diferentes clasificadores para lograr una mayor precisión (Cui et al., 2016). La hipótesis en que se basa este tipo de métodos plantea que se puede mejorar el rendimiento de la clasificación de series de tiempo a través del ensamblaje de un conjunto homogéneo o heterogéneo de clasificadores, que se adapten a las características implícitas de las clases de una serie de tiempo.

El conjunto elástico (PROP) (Bagnall, 2014) combina 11 clasificadores basados en medidas de distancia elástica con un esquema ponderado de conjunto .

Shapelet ensemble (SE) (Bagnall et al., 2015) produce los clasificadores a través de la transformada shapelet¹ junto con un conjunto heterogéneo.

El colectivo plano de conjuntos basados en transformaciones, COTE (Bagnall et al., 2015), es un conjunto de 35 clasificadores diferentes proporcionales a su entrenamiento de precisión de validación cruzada, basados en las características extraídas tanto de los dominios de tiempo como de frecuencia.

A pesar de su alta precisión de prueba obtenida mediante la combinación de clasificadores, los métodos basados en conjuntos de modelos sufren una alta complejidad como consecuencia intrínseca de su naturaleza.

¹ Shapelet es utilizado para formar las reglas dentro de un árbol de decisión

3.2 APRENDIZAJE PROFUNDO, CLASIFICACIÓN DE SERIES DE TIEMPO Y OTRAS PROBLEMATICAS

Recientemente, enfoques de aprendizaje profundo han sido utilizados para problemas de clasificación de series de tiempo, esto dado el éxito que ha tenido este tipo de enfoques en otros problemas de clasificación (Wang et al., 2016). A pesar de ello, en la literatura reciente se puede evidenciar que no se ha indagado mayormente en la posibilidad que ofrecen los métodos de aprendizaje profundo para resolver los problemas de TSC (Fawaz et al., 2019). Es así como en Bagnall et al. (2017), se realiza la evaluación de 18 métodos recientes de TSC, sin considerar ninguno con un enfoque en aprendizaje profundo.

Es por ello, que se debe resaltar que la principal ventaja que presenta este enfoque, es que los métodos no necesitan una ingeniería de características previas del problema que se aborda, sino que van diseñando sus propias características durante el entrenamiento. A partir de ello, dentro de otras problemáticas, algunos ejemplos se destacan en el desempeño de las CNN en reconocimiento y clasificación de imágenes, y en el uso de otras arquitecturas profundas para resolver tareas relacionadas al procesamiento de lenguaje natural, como la traducción automática o la clasificación de documentos, o en su uso para reconocimiento de voz (Fawaz et al., 2019).

Dicho éxito se demuestra en que los resultados teóricos reflejan que para el aprendizaje adecuado de funciones complejas asociadas a tareas con alto nivel de abstracción, es necesario utilizar arquitecturas profundas (Bengio, 2009).

En lo que respecta a los estudios que se han realizado para los problemas de clasificación de series de tiempo con enfoques de aprendizaje profundo, en Wang et al. (2016) y Geng & Luo (2019), es diseñado un perceptrón multicapa (MPL) para aprender desde cero un clasificador discriminatorio de series de tiempo. El problema que conlleva trabajar con una MLP, es que la información temporal se pierde y las características aprendidas ya no son invariantes en el tiempo. Aquí se decantan otro tipo de modelos tales como las LSTM que fueron diseñadas especialmente para tratar con series de tiempo aprendiendo dinámicamente de las dependencias a largo plazo a través de los carruceles de error constante (Bakker, 2003), así como las CNN, que pueden aprender a través de filtro espacialmente invariantes de las series de tiempo de entrada sin procesar (Wang et al., 2016), (Geng & Luo, 2019). Precisamente el enfoque mayormente usado para afrontar los problemas de TSC son las CNN.

Particularmente, un estudio interesante se realiza en Wang et al. (2016), en el cual se lleva a cabo una análisis comparativo de una CNN totalmente conectada (fully convolutional network, FCN), una MLP (antes mencionada) y una ResNet (Residual Networks). En el caso de las FCN, estas son principalmente redes convolucionales que no contienen ninguna capa de agrupación local, lo que significa la serie de tiempo se mantiene sin cambios a lo largo de las convoluciones, y cambian la capa final por una capa de Agrupación Promedio Global (GAP), que reduce drásticamente el

número de parámetros en una red neuronal. Las ResNet por otra parte son redes residuales relativamente profundas en las que se agrega un atajo lineal para vincular la salida de un bloque residual a su entrada, lo que permite el flujo del gradiente directamente a través de estas conexiones, lo que hace que la capacitación de una red neuronal profunda sea más fácil al reducir el efecto de desvanecimiento de la gradiente (He et al., 2016).

En Cui et al. (2016), se utiliza un modelo end-to-end de una CNN a multiescala, muy similar a un modelo tradicional, pero que agrega el método Window Slicing (WS) como una técnica de aumento de datos. WS desliza una ventana sobre la serie de tiempo de entrada y extrae las subsecuencias, por lo que entrena a la red en las subsecuencias extraídas en lugar de las series de tiempo de entrada sin procesar.

En Zheng et al. (2016) se propone una CNN multicanal en el que las convoluciones se aplican de forma independiente, en paralelo en cada dimensión o canal de la serie de tiempo.

En Zhao et al. (2017) se propone un enfoque de time-CNN, en el que a diferencia de un modelo tradicional se utiliza el error cuadrático medio (MSE) en lugar de la función de pérdida de entropía cruzada categórica tradicional, y además se utiliza una capa final clasificadora con sigmoide como la función de activación, no garantizando una suma de probabilidades igual a 1, y está conectada directamente a la salida de la anterior convolución, sin una capa intermedia.

Todos estos estudios se enmarcan dentro de la experimentación con el repositorio de series de tiempo de la Universidad de California (UCR).

Por otro lado, en el dominio de la bioinformática, en el que despierta gran interés el análisis de series de tiempo, se destaca que las CNN son el método dentro del aprendizaje profundo más ampliamente utilizado para la clasificación de señales fisiológicas (Faust et al., 2018). Ejemplo de ello es que para detectar infracciones de miocardio a partir de datos de electrocardiografía Strodthoff & Strodthoff (2019) propusieron un método utilizando CNN profundas. Así mismo, Liu et al. (2018) aprovechó un modelo de CNN para características multivariadas y de características de retraso con el fin de lograr una precisión de vanguardia en los datos del desafío 2015 de Pronósticos y Gestión de la Salud (PHM).

3.2.1 Arquitecturas híbridas de aprendizaje profundo

Un último enfoque que ha proliferado se remarca en la combinación de distintos métodos de aprendizaje profundo, a lo que se puede denominar como desarrollos de "arquitecturas híbrida". Entre este tipo de arquitecturas, principalmente se despliega el uso de las CNN en combinación con otro tipo de métodos. Algunos ejemplos son la combinación entre CNN y las unidades recurrentes cerradas (Lin & Runger, 2018), y entre CNN y un mecanismo de atención convolucional para la construcción de un encoder para series de tiempo (Serrà et al., 2018).

Así mismo, en (Shi et al., 2015), se propone un modelo covolucional totalmente conectado LSTM

(FC-LSTM), que posee estructuras convolucionales en las transiciones de entrada a estado y estado a estado, utilizando el LSTM convolucional (ConvLSTM) para construir un modelo de extremo a extremo entrenable para el problema de difusión de precipitación, que busca predecir la intensidad de la precipitación futura en una región local durante un período de tiempo relativamente corto.

Otro ejemplo se enmarca en el uso de una red neuronal convolucional y una red de autopistas sobre caracteres, cuya salida se otorga a una red neuronal recurrente LSTM (RNN-LM), para el modelado de idiomas en base a caracteres (Kim et al., 2012).

Así como estos ejemplos, en la literatura se pueden encontrar diversos usos de arquitecturas híbridas convolucionales recurrentes que son entrenables de extremo a extremo y adecuadas para tareas de comprensión visual a gran escala, reconocimiento de actividades, subtítulos de imágenes y descripción de videos, donde las CNN se utilizan como extractores de características para los LSTM en datos de entrada de audio y de texto (Donahue et al., 2017)

CAPÍTULO 4. MÉTODOS

4.1 MÉTODO

Para llevar a cabo el desarrollo de un modelo de clasificación de series de tiempo basado en una arquitectura híbrida de aprendizaje profundo, se propone implementar un método de 4 etapas, las cuales se nombran a continuación:

1. **Datos:** Selección de los conjuntos de datos de series de tiempo. Para esto se usa como conjunto de referencia al repositorio de la Universidad de California para problemas de TSC.
2. **Modelos de aprendizaje profundo:** Se desarrollan diferentes implementaciones de arquitecturas de aprendizaje profundo, mediante la combinación de redes neuronales recurrentes LSTM y redes neuronales convolucionales CNN.
3. **Selección de los modelos:** Se comparan los diversos modelos implementados mediante un test de hipótesis verificando si existen diferencias significativas entre los modelos implementados. De ser así, se implementa un test post-hoc que junto a otras herramientas permita distinguir aquellos modelos que difieran entre sí. Para la selección final del modelo se toma apoyo de la evaluación general con distintas métricas a manera de establecer el mejor modelo implementado.
4. **Comparación con otros modelos:** A partir de la selección del mejor modelo se establece una comparación estadística con otros modelos implementados bajo el enfoque de aprendizaje profundo, así como otros modelos de mejor rendimiento presentes en la literatura. Cabe destacar que dicha comparación se establece en términos de los resultados que se pueden encontrar en los diversos papers que describen el desarrollo de estos modelos, y no se establece desde una base empírica propia.

4.2 CONJUNTOS DE DATOS

Los conjuntos de datos considerados para poner a prueba las arquitecturas híbridas desarrolladas, provienen del Repositorio de la Universidad de California (UCR) para la clasificación de series de tiempo. Este repositorio cuenta con amplios conjuntos de series de tiempo provenientes de diversas problemáticas y ha sido la base para las últimas investigaciones dentro de este ámbito (Dau et al., 2015). Es así como en la literatura se pueden encontrar gran variedad de métodos que realizan su experimentación con conjuntos de datos de dicho repositorio, con lo que es posible establecer un análisis comparativo con el método aquí propuesto.

La lista de los conjuntos de datos considerados se compone de 44 en total. Se

seleccionaron en base a los resultados que exponen la literatura de su experimentación con diversos métodos de TSC. A continuación se presenta la lista con una breve descripción de la problemática que aborda.

- **Adiac:** El proyecto ADIAC para la identificación y clasificación automáticas de diatomeas, un tipo de algas celulares que constituye uno de los tipos más comunes de filoplanton, fue un estudio piloto sobre la identificación automática sobre la base de imágenes. Los contornos se extraen de imágenes de umbral, en lo que presumiblemente, se generan series de tiempo como distancia a un punto de referencia.
- **Beef:** Los espectrógrafos de alimentos se utilizan en quimiometría para clasificar los tipos de alimentos, una tarea que tiene aplicaciones en la seguridad de los alimentos y control de calidad. El conjunto de datos Beef considera datos en cuatro clases de espectrogramas de carne de res, de carne de vaca pura y carne de res adulterada con diversos grados de despojos, para su clasificación.
- **CBF:** Cylinder-Bell-Funnel es un conjunto de datos simulados definidos por N. Saiko en su tesis "Extracción de características locales y sus aplicaciones utilizando una biblioteca de bases". Los datos de cada clase son ruido normal estándar más un término de compensación que difiere para cada clase.
- **ChlorineCon:** Este conjunto de datos fue definido en una tesis doctoral por Lei Li (Carnegie Mellon University). Fue producido por EPANET que modela el comportamiento hidráulico y de calidad del agua de los sistemas de tuberías de distribución de agua. EPANET puede rastrear, en una red de agua dada, el nivel de agua y la presión en cada tanque, el flujo de agua en las tuberías y la concentración de una especie química (cloro en este caso) en toda la red dentro de una duración simulada. El conjunto de datos consta de 166 nodos (uniones de tubería) y medición del nivel de concentración de cloro en todos estos nodos durante 15 días (una medición por cada 5 minutos, un total de 4310 tics de tiempo).
- **CinCECGTorso:** Estos datos se derivan de uno de los desafíos computacionales en cardiología, una competencia anual en physionet. Los datos se toman de los datos de electrocardiografía (ECG) para múltiples sitios de superficie del torso. Hay 4 clases (4 personas diferentes).
- **Coffee:** El conjunto de datos de coffee cuyos datos provienen de distintas spectrograffías, es un problema de dos clases para distinguir entre los granos de café Robusta y Aribica.
- **CricketX, CricketY y CricketZ:** Las series Cricket X, Y y Z son datos de acelerómetro (en tres dimensiones) tomados de actores que realizan gestos de cricket. Las doce clases son

diferentes señales que puede emitir un árbitro. Montaron dos acelerómetros ortogonales entre sí, por lo que la aceleración se mide en el espacio 3D. En su conjunto representan una serie de tiempo multivariable.

- **DiatomSizeReduction:** Conjunto de datos de los esquemas de cuatro tipos de diatomeas, Gomphonema augur, Fragilariforma bicapitata, Stauroneis smithii y Eunotia tenella. Existe una variación de tamaño dentro de la clase porque cada generación sucesiva de una diatomea de reproducción clonal es ligeramente más pequeña que sus antepasados.
- **ECGFiveDays:** Conjunto de datos proveniente de la electrocardiografía de un hombre de 67 años de edad. Presenta dos clases que son simplemente representaciones en tiempo de la ECG.
- **FaceAll:** Conjunto de datos provenientes de la representación de rostros pertenecientes a 14 estudiantes. Cada contorno facial está mapeado en una serie unidimensional.
- **FaceFour:** Datos de un problema de contorno de clase de cuatro clases. No se tiene mayor información de su procedencia.
- **FacesUCR:** Conjunto de datos, versión de FaceAll alineada por rotación con una división de prueba/entrenamiento diferente.
- **Fiftywords:** Conjunto de datos de contornos de palabras extraídos de la biblioteca de George Washington por T. Rath. Cada caso es una palabra. Una serie se forma tomando el perfil de altura de la palabra.
- **fish:** Un conjunto de contornos de peces utilizados para el reconocimiento de especies de peces y monitoreo de la migración. Cada clase es una especie diferente.
- **GunPoint:** Este conjunto de datos involucra a una actriz y un actor masculino que hacen un movimiento con la mano. Para las clases, se rastrea el centroide de las manos derechas del actor en los ejes X e Y, que parecen estar altamente correlacionados. Los datos en el archivo son solo el eje X.
- **Haptics:** Datos tomados de 5 personas que ingresan su contraseña (un código para acceder a un sistema protegido por un sistema de autenticación gráfica) en una pantalla táctil. El dato es solo el movimiento del eje X.
- **InlineSkate:** Datos recopilados de pruebas en patinaje de velocidad en línea profesional. El atleta realizó una prueba de interior estandarizada a una velocidad de 7,89 m/s en una cinta rodante motorizada grande. Los datos de EMG y cinemáticos del lado derecho del cuerpo se midieron durante 30 segundos, correspondientes a 19 ciclos de movimiento completos.

Los datos de EMG de 7 músculos extensores y flexores principales de la pierna del lado derecho del cuerpo se recolectaron utilizando electrodos bipolares de superficie unidos a los músculos relevantes. Para estos datos, cada serie de tiempo representa una medida angular del tobillo durante un ciclo de movimiento. Los datos se han truncado a igual longitud.

- **ItalyPowerDemand:** Datos derivados de doce series mensuales de demanda de energía eléctrica de Italia. La tarea de clasificación es distinguir los días de octubre a marzo (inclusive) de abril a septiembre.
- **Lightning2:** Datos provenientes del satélite FORTE detecta los eventos electromagnéticos transitorios asociados con los rayos utilizando un conjunto de instrumentos ópticos y de radiofrecuencia (RF). Los datos se recopilan con una frecuencia de muestreo de 50 MHz durante 800 microsegundos. Se realiza una transformada de Fourier en los datos de entrada para producir un espectrograma. Los espectrogramas luego se colapsan en frecuencia para producir una serie de tiempo de densidad de potencia, con 3181 muestras en cada serie de tiempo. Luego se suavizan para producir series de longitud 637.
- **Lightning7:** Datos también provenientes del satélite FORTE.
- **Mallat:** Conjunto de datos simulado.
- **MedicalImages:** Datos procedentes de histogramas de la intensidad de píxel de diversas imágenes médicas. Las clases son diferentes regiones del cuerpo humano.
- **MoteStrain:** Los datos del sensor originalmente de Carlos Guestrin (CMU). La tarea es distinguir entre el sensor q8calibHumid (una medida de humedad) y el sensor q8calibHumTemp (temperatura). Los datos tienen dropouts. En los datos originales, los dropouts tienen un valor de 0. Esto significa que son difíciles de detectar con datos normalizados.
- **NonInvThorax1:** Conjunto de datos que corresponde al registro de un ECG de tórax izquierdo y derecho de un feto. Cada tipo de ECG fetal no invasivo corresponde a una serie de tiempo 2D.
- **NonInvThorax2:** Simil a NonInvThorax1.
- **OliveOil:** Escpectografía para la quimiometría de un aceite de oliva virgen extra de países alternativos. La espectroscopía FTIR y el análisis multivariado pueden distinguir el origen geográfico de los aceites de oliva virgen extra.
- **OSULeaf:** Conjunto de datos que registra contornos unidimensionales de hojas. Las series se obtuvieron mediante segmentación de imágenes en color y extracción de límites (en

sentido contrario a las agujas del reloj) a partir de imágenes de hojas digitalizadas de seis clases: Acer Circinatum, Acer Glabrum, Acer Macrophyllum, Acer Negundo, Quercus Garryana y Quercus Kelloggii.

- **SonyAIBORobot:** Conjunto de datos procedentes de un robot con acelerómetros de balanceo/cabeceo/ desvío. Este dato es solo el eje X La tarea es detectar la superficie sobre la que se camina (cemento o alfombra para Sony1).
- **SonyAIBORobotII:** Simil a SonyAIBORobot.
- **StarLightCurves:** Agrupación de la luz estelar. Una curva de luz estelar es una serie de tiempo del brillo de un objeto celeste en función del tiempo. El estudio de las curvas de luz en astronomía está asociado con el estudio de la variabilidad de las fuentes. Este es un conjunto de datos con 1,000 curvas de luz estelar alineadas en fase de longitud 1,024, cuya clase ha sido determinada por un experto.
- **SwedishLeaf:** Conjunto de contornos de hojas de árboles suecos. En total el conjunto de datos posee 15 clases distintas.
- **Symbols:** Datos provenientes de un experimento con 13 personas, a quienes se les pidió que copiaran el símbolo que aparecía al azar lo mejor que pudieron. Hubo 3 símbolos posibles, cada persona contribuyó con unos 30 intentos. Los datos son el movimiento del eje X para dibujar la forma.
- **SyntheticControl:** Conjunto de datos que contiene 600 ejemplos de gráficos de control generados sintéticamente por el proceso en las consultas similares de similitud de series temporales que emplean un enfoque basado en características. Hay seis clases diferentes de gráficos de control: 1. Normal 2. Cíclico 3. Tendencia creciente 4. Tendencia decreciente 5. Desplazamiento hacia arriba 6. Desplazamiento hacia abajo
- **Trace:** Este conjunto de datos de 4 clases es un subconjunto del Índice de Clasificación de Transitorios, una iniciativa en el cambio de siglo para recopilar datos del dominio de aplicación de la industria de procesos (por ejemplo, nuclear, química, etc.). Es un conjunto de datos sintéticos diseñado para simular fallas de instrumentación en una planta de energía nuclear.
- **TwoLeadECG:** Conjunto de datos de ECG tomado de physionet. Específicamente, los datos provienen de la base de datos de ECG a largo plazo MIT-BIH (ltdb), registro ltdb/15814, que comienzan en el tiempo 420 y terminan en 1019. La tarea es distinguir entre la señal 0 y la señal 1.

- **TwoPatterns:** Conjunto de datos simulado generado por P. Geurts para su tesis doctoral de 2002 Contribuciones a la inducción del árbol de decisión: compensación de sesgo / varianza y clasificación de series de tiempo. de la Universidad de Lieja.
- **UWaveX, UWaveY y UWaveZ:** Conjunto de ocho gestos simples generados a partir de acelerómetros. Reconocimiento de gestos personalizado basado en acelerómetro y sus aplicaciones. Los datos consisten en las coordenadas X, Y, Z de cada movimiento.
- **Wafer:** Datos de las obleas de silicio consistentes a la fabricación de microelectrónica de semiconductores. Cada conjunto de datos en la base de datos de obleas contiene las mediciones registradas por un sensor durante el procesamiento de una oblea por una herramienta.
- **WordSynonyms:** Este conjunto de datos consta de perfiles de palabras para los manuscritos de George Washington. Consta de 50 palabras remapeadas a 25 clases. Los datos se voltearon de izquierda a derecha para que no fueran reconocidos.
- **Yoga:** Conjunto de datos obtenidos de las capturas a dos actores que transitan entre posturas de yoga frente a una pantalla verde. El problema es discriminar entre un actor (hombre) y otro (mujer). Cada imagen se convirtió en una serie unidimensional encontrando el contorno y midiendo la distancia del contorno al centro.

En lo que respecta a las características de tamaño, cantidad de clases y tipo de proveniencia, estas son presentadas en la Tabla 4.1.

4.3 MODELOS DE APRENDIZAJE PROFUNDO

Para la búsqueda de un buen modelo clasificador de series de tiempo, se desarrollan distintas arquitecturas en base a las redes neuronales convoluciones CNN y las redes recurrentes LSTM. A continuación, se abordan las arquitecturas escogidas, las técnicas utilizadas para el entrenamiento, y la configuración de los hiper-parámetros de cada red neuronal.

4.3.1 Desarrollo de las arquitecturas

Se entrenaron ocho arquitecturas de aprendizaje profundo, construidas empíricamente en base a la combinación de los dos tipos de redes neuronales consideradas. La combinación se da en base a el orden en cuanto a su estructura, como en la cantidad de capas convolucionales y LSTM consideradas para cada red.

Además de las capas ya mencionadas, se utilizan las siguientes capas:

- **Dropout:** Establece aleatoriamente una tasa de fracción de unidades de entrada en 0 en cada actualización durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste. En esta ocasión la tasa de fracción es definida en 0,5

Tabla 4.1: Características de los conjuntos de datos seleccionados.

Conjunto de datos	Cantidad de datos	Lomgitud	No. de Clases	Tipo
CinCECGtorso	1420	1639	4	ECG
ECGFiveDays	884	136	2	
NonInvFetalECGThorax1	3765	750	42	
NonInvFetalECGThorax2	3765	750	42	
TwoLeadECG	1162	82	2	
Adiac	781	176	37	
DiatomSizeReduction	322	345	4	IMAGE
FaceAll	2250	131	14	
FaceFour	112	350	4	
FacesUCR	2250	131	14	
FiftyWords	905	270	50	
Fish	350	463	7	
MedicallImages	1141	99	10	
OSULeaf	442	427	6	
SwedishLeaf	1125	128	15	
Symbols	1020	398	6	
WordSynonyms	905	270	25	MOTION
Yoga	3300	426	2	
CricketX	780	300	12	
CricketY	780	300	12	
CricketZ	780	300	12	
GunPoint	200	150	2	
Haptics	463	1092	5	
InlineSkate	650	1882	7	
UWaveGestureLibraryX	4478	315	8	
UWaveGestureLibraryY	4478	315	8	
UWaveGestureLibraryZ	4478	315	8	
ItalyPowerDemand	1096	24	2	SENSOR
Lightning2	121	637	2	
Lightning7	143	319	7	
MoteStrain	1272	84	2	
SonyAIBORobotSurface1	6211	70	2	
SonyAIBORobotSurface2	980	65	2	
StarlightCurves	9236	1024	3	
Trace	200	275	4	
Wafer	7164	152	2	SIMULATED
CBF	930	128	3	
ChlorineConcentration	4307	166	3	
Mallat	2400	1024	8	
SyntheticControl	600	60	6	
TwoPatterns	5000	128	4	
Beef	60	470	5	SPECTRO
Coffee	56	286	2	
OliveOil	60	570	4	

Fuente: Elaboración propia, (2018)

- **MaxPooling:** Reduce la resolución de los mapas de características obtenidos en las capas convolucionales anteriores, extrayendo el máximo valor de las características.

- **GlobalAveragePooling:** Al igual que MaxPooling, reduce la resolución de los mapas de características obtenidos en las capas *convolucionales* anteriores, extrayendo el valor promedio de las características.
- **FullyConnected:** Capa en la que cada neurona está conectada a cada neurona en la capa anterior, y cada conexión tiene su propio peso. Este es un patrón de conexión de propósito general y no hace suposiciones sobre las características en los datos. Es la antesala de la capa de salida.
- **Flatten:** Aplana las dimensiones de la salida anterior generando un vector de una dimensión.
- **Capa de salida:** Capa que implementa una función Softmax para transformar todas las activaciones netas en la capa de salida final en una serie de valores que pueden interpretarse como probabilidades, las que al final serán las probabilidades de salida para cada clase.

Arquitecturas profundas

Cada una de las arquitecturas desarrolladas se ilustran a continuación:

CNN-BiLSTM: Arquitectura compuesta por dos capas *convolucionales* de una dimensión, seguidas de una capa de *Dropout*, una capa de *MaxPooling*, una capa *BiLSTM* con 100 unidades de salida, para seguir con una capa *FullyConnected* y posteriormente con la capa de salida con una función *Softmax* para la clasificación.

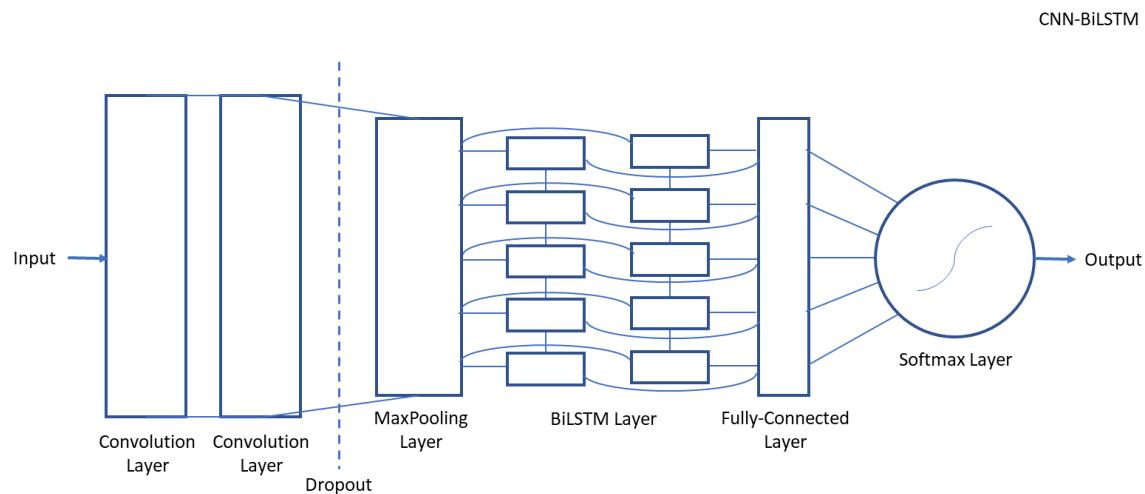


Figura 4.1: Arquitectura 1, CNN-BiLSTM.
Fuente: Elaboración propia, (2018).

LSTM-CNN: Arquitectura compuesta por una capa *LSTM* con 100 unidades de salida, seguido de una capa *convolucional* de una dimensión, una capa de *MaxPooling*, una capa *Dropout*, una capa *Flatten*, una capa *FullyConnected* y posteriormente con la capa de salida con una función *Softmax* para la clasificación.

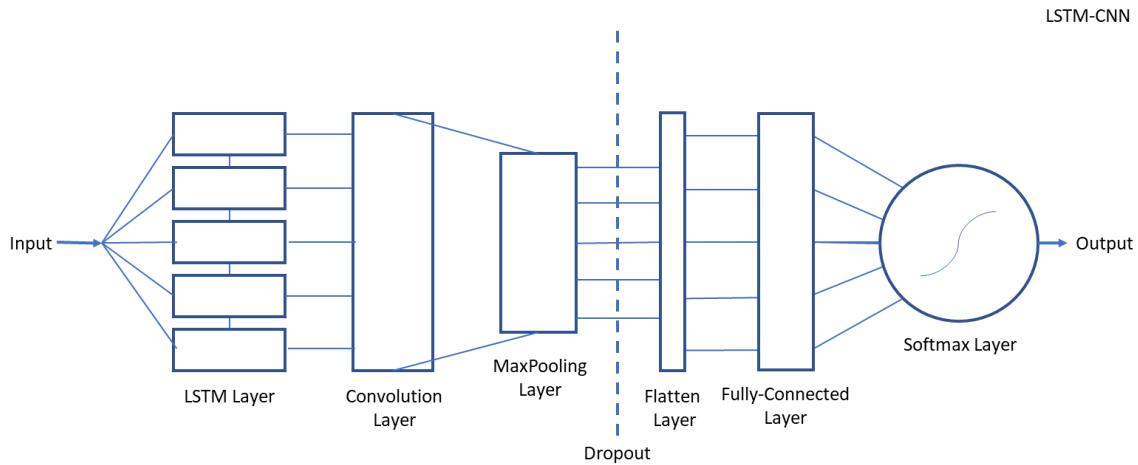


Figura 4.2: Arquitectura 2, LSTM-CNN.

Fuente: Elaboración propia, (2018).

CNN-LSTM-Separated: Arquitectura compuesta por dos raíces. La primera raíz, se compone de 3 capas *convolucionales*, una capa *Dropout*, una capa *MaxPooling*, y una capa *Flatten*, mientras que en la otra raíz, se tiene una capa *LSTM* con 100 unidades de salida y una capa *Dropout*. Las dos raíces se concatenan y pasan a una capa *FullyConnected* y posteriormente a la capa de salida con *Softmax* para la clasificación.

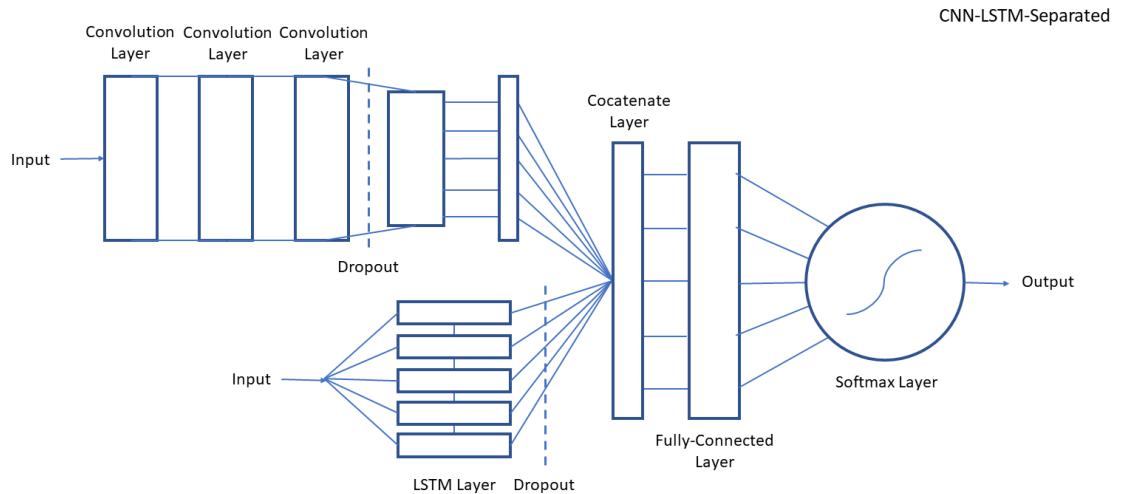


Figura 4.3: Arquitectura 3, CNN-LSTM-Separated.
Fuente: Elaboración propia, (2018).

Multi-CNN-BiLSTM-2: Arquitectura compuesta por dos raíces. Ambas raíces completamente iguales, compuestas de 2 capas *convolucionales*, una capa *Dropout*, una capa *MaxPooling*, y una capa *Flatten*. Las dos raíces se concatenan y pasan a una capa *BiLSTM* con 100 unidades de salida, para seguir a una capa *FullyConnected* y posteriormente a la capa de salida con *Softmax* para la clasificación.

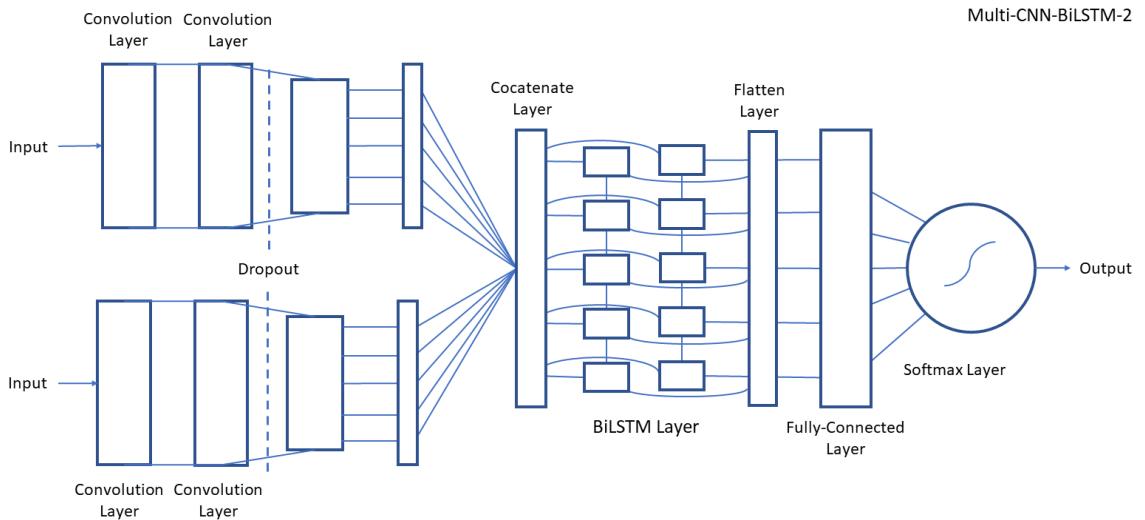


Figura 4.4: Arquitectura 4, Multi-CNN-BiLSTM-2.
Fuente: Elaboración propia, (2018).

Multi-CNN-BiLSTM: Arquitectura compuesta por tres raíces, exactamente iguales, las cuales componen de 2 capas *convolucionales*, una capa *Dropout*, una capa *MaxPooling*, una capa *BiLSTM* con 100 unidades de salida y una capa *Flatten*. Las tres raíces se concatenan y

pasan a una capa *FullyConnected*, para luego finalizar con la capa de salida con *Softmax* para la clasificación.

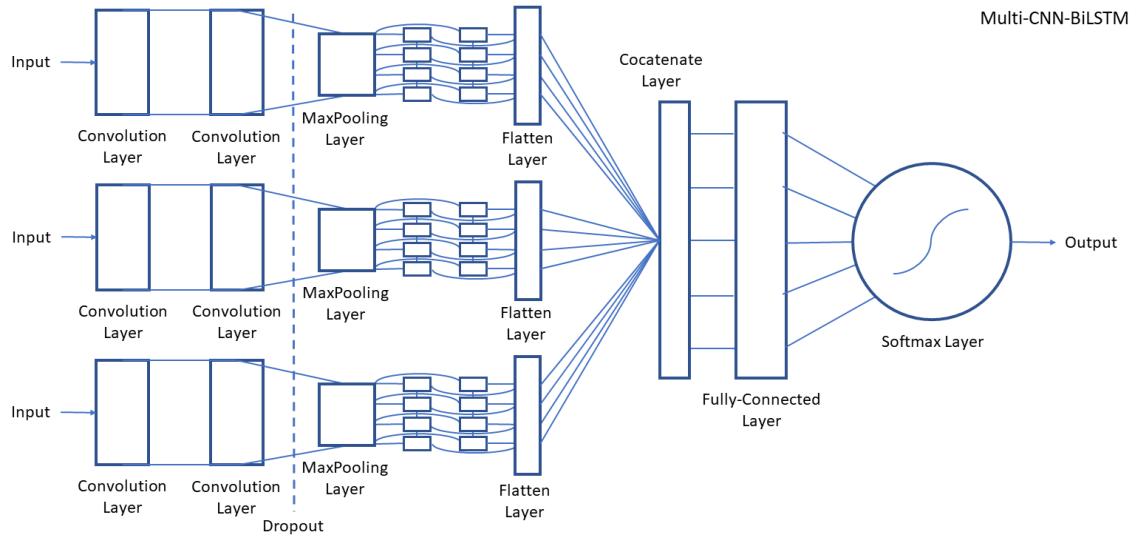


Figura 4.5: Arquitectura 5, Multi-CNN-BiLSTM.
Fuente: Elaboración propia, (2018).

CNN-LSTM-Separated-2: Arquitectura compuesta por tres raíces. La primera raíz, se compone de 3 capas *convolucionales*, una capa *Dropout*, una capa *MaxPooling*, y una capa *Flatten*. La segunda raíz se compone de 2 capas *convolucionales*, 1 capa *MaxPooling* y una capa *LSTM*, mientras que en la otra raíz, se tiene una capa *LSTM* con 100 unidades de salida y una capa *Dropout*. Las tres raíces se concatenan y pasan a una capa *FullyConnected*, para luego finalizar con la capa de salida con *Softmax* para la clasificación.

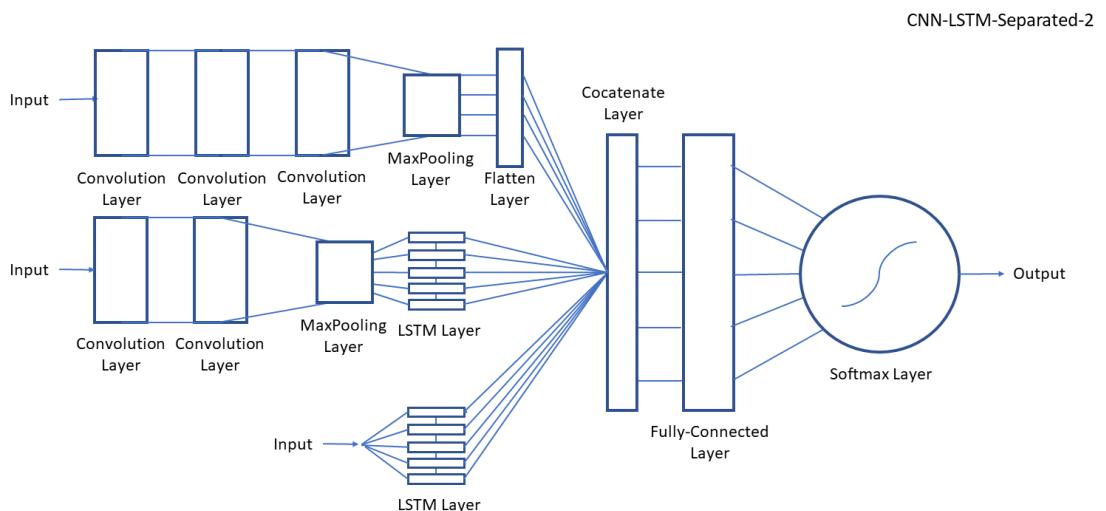


Figura 4.6: Arquitectura 6, CNN-LSTM-Separated-2.
Fuente: Elaboración propia, (2018).

FCN-BiLSTM: Arquitectura basada en el modelo de Fully Convolutional Network (FCN) agregando una capa *BiLSTM* con 100 unidades de salida después del bloque convolucional y una capa *Flatten* antes de la capa de salida.

FCN-BiLSTM

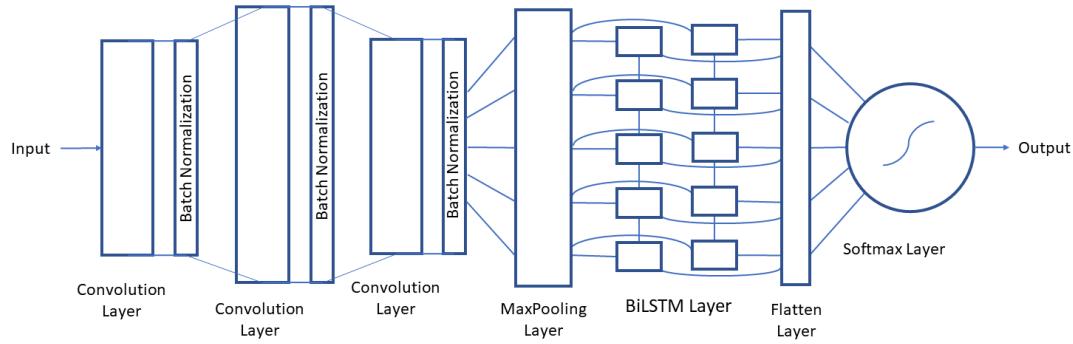


Figura 4.7: Arquitectura 7, FCN-BiLSTM.
Fuente: Elaboración propia, (2019).

BiLSTM-FCN: Arquitectura basada en el modelo de Fully Convolutional Network (FCN) agregando una capa *BiLSTM* con 100 unidades de salida antes del bloque convolucional.

BiLSTM-FCN

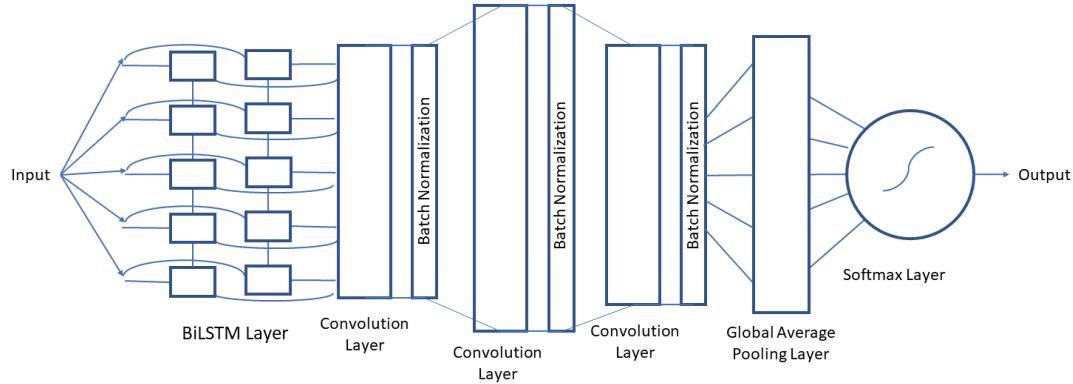


Figura 4.8: Arquitectura 8, BiLSTM-FCN.
Fuente: Elaboración propia, (2019).

Resnet-BiLSTM: Arquitectura basada en el modelo de Residual Networks (Resnet) agregando una capa *BiLSTM* con 100 unidades de salida después del bloque convolucional y una capa

Flatten antes de la capa de salida.

Resnet-BiLSTM

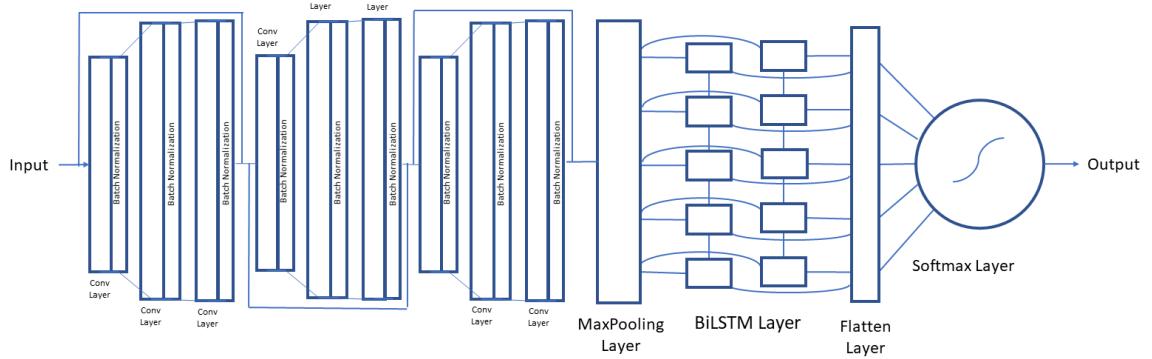


Figura 4.9: Arquitectura 9, Resnet-BiLSTM.

Fuente: Elaboración propia, (2019).

BiLSTM-Resnet: Arquitectura basada en el modelo de Residual Networks (Resnet) agregando una capa *BiLSTM* con 100 unidades de salida antes del bloque convolucional.

BiLSTM-Resnet

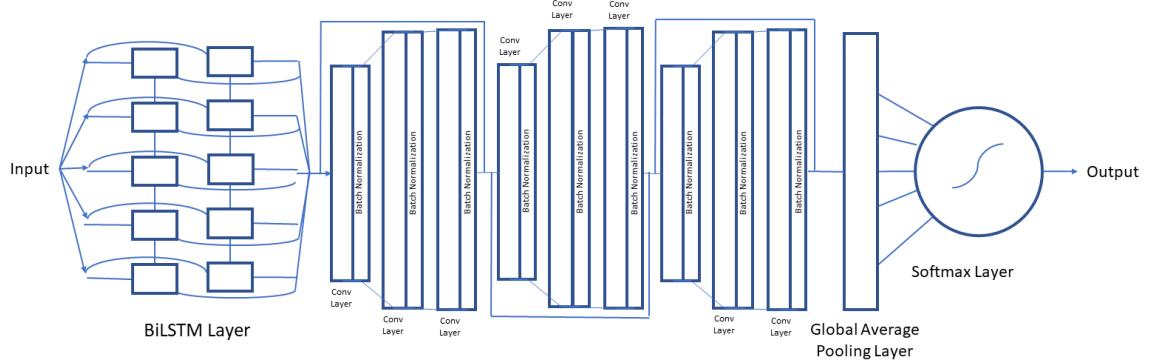


Figura 4.10: Arquitectura 10, BiLSTM-Resnet.

Fuente: Elaboración propia, (2019).

Finalmente un resumen de los componentes e de cada una de las arquitecturas implementadas, se presenta a continuación:

Tabla 4.2: Componentes de las arquitecturas implementadas.

	#Capas	#Conv	#LSTM	Normalización	Pooling	Características	Regularización
Istm_cnn	6	2	1-Bi	No	Max	FC	Dropout
cnn_bilstm	6	2	1-Bi	No	Max	FC	Dropout
cnn_lstm_separated	9	3	1	No	Max	FC	Dropout
multi_cnn_bilstm2	13	4	1-Bi	No	Max	FC	Dropout
cnn_lstm_separated2	13	4	2	No	Max	FC	Dropout
multi_cnn_bilstm	15	6	3-Bi	No	Max	FC	Dropout
FCN-BiLSTM	10	3	1-Bi	Si	Max	FC	No
BiLSTM-FCN	9	3	1-Bi	Si	No	GAP	No
Resnet-BiLSTM	22	9	1-Bi	Si	Max	FC	No
BiLSTM-Resnet	21	9	1-Bi	Si	No	GAP	No

Fuente: Elaboración propia, (2019).

Hiperparámetros

Con cada una de las arquitecturas, se realizó una grilla de parámetros, entre los cuales se variaron los siguientes aspectos:

- Número de filtros de las capas convolucionales.
- Cantidad de épocas.
- Tamaño del Batch.
- Cantidad de bloques LSTM.
- Función de optimización.
- Función objetivo.
- Función de optimización.

Si bien en la experimentación se denota que la grilla de parámetros en todos los modelos difiere en cada conjunto de datos, se optó porque el modelo fuera el mismo para todos los conjuntos de datos, tomando en consideración que el objetivo de este proyecto se enmarca en encontrar un modelo de clasificador general y no específico.

La configuración de hiperparámetros utilizada en general para cada uno de los modelos implementados se resume en la tabla a continuación:

Técnicas utilizadas para el entrenamiento

Preprocesamiento de los datos Es común que en los datos se encuentren características cuyos valores se encuentren en diferentes escalas. Esto en el proceso de aprendizaje de una red neuronal puede provocar que ciertos pesos pueden actualizarse más rápido que otros ya que los valores de las características juegan un papel en las actualizaciones

Tabla 4.3: Configuración de hiperparámetros para el entrenamiento de los modelos.

Hiperparámetro	Grilla de Parámetros	Parámetro considerado
Cantidad de filtros capas convolucionales	{64, 32, 16}	64
Bloques LSTM	{50, 100}	100
Cantidad de épocas	2500	Depende del conjunto de datos.
Tamaño del batch	{64, 32}	64
Función objetivo	Categorical cross entropy, Cross entropy	Categorical loss Entropy
Función de activación	ReLU, Sigmoide	ReLU
Función de optimización	Adam, RMSProp	Adam

Fuente: Elaboración propia, (2019).

de peso. Es por ello que es necesario realizar un preprocesamiento de los datos que estandarice el rango de características. Las opciones consideradas son el escalamiento, la normalización y la estandarización.

Por un lado, el escalamiento transforma los datos de modo que las características estén dentro de un rango específico, por ejemplo, [0, 1].

$$x_{scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

Es útil en el procesamiento de imágenes.

En el caso de la normalización, esta busca que los datos puedan describirse como una distribución normal.

$$x_{normalized} = \frac{x - mean(x)}{max(x) - min(x)}$$

Es útil cuando se aplican técnicas estadísticas que asuman que los datos se distribuyen normalmente.

Y en el caso de la estandarización o normalización en Z, los datos se transforman de manera que la distribución resultante tenga una media de 0 y una desviación estándar de 1.

$$x_{standarized} = \frac{x - \mu}{\sigma}$$

Finalmente se optó por la estandarización dado que la normalización es sensible a los valores atípicos, y el escalamiento Además la estandarización a veces puede mejorar la velocidad de convergencia del algoritmo en el descenso de gradiente.

Épocas de entrenamiento Todos los modelos implementados fueron entrenados con 2500 épocas. Como todos los conjuntos de datos son diferentes entre si, se utilizó la técnica

de Early Stopping buscando evitar el sobreajuste de los modelos, considerando que 2500 épocas es un valor alto. Ante el uso de esta técnica, para verificar que el modelo generalizara lo suficientemente bien y fuera eficiente en cuanto a las épocas de entrenamiento consideradas, se utilizó una espera de 200 épocas una vez encontrado el mejor modelo. De esta manera se alcanza mayor generalización sin detener el modelo muy tempranamente. Esto resulta en que dada x como la época en donde se encuentra el mejor modelo, si en $x + i$, con $i = 1, 2, 3, \dots, 200$, no se encuentra un modelo mejor, entonces el entrenamiento se detiene; en otro caso x_{new} pasa a ser la época en donde se encuentra un nuevo mejor modelo y se repite el proceso anterior.

Validación cruzada Se utiliza validación cruzada para obtener un modelo con mejor capacidad de generalización ante nuevos datos. Se consideró 5-fold en general para todos los modelos.

Métricas de análisis de los modelos

En general las métricas consideradas para el análisis de los distintos modelos desarrollados contemplan el accuracy y el error de entropía cruzada, el coeficiente Kappa, y el área bajo la curva ROC.

El accuracy, es la proporción de observación pronosticada correctamente con respecto al total de observaciones.

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

Esta medida es significante sólo cuando se tienen conjuntos de datos simétricos donde los valores de falso positivo y falso negativo son similares. Por lo tanto, es recomendable observar otras métricas para evaluar el rendimiento de los modelos.

La entropía cruzada tal como se trata en el capítulo 2, es una medida de error aumenta cuando la probabilidad predicha difiere de la etiqueta real, penalizando las predicciones que son confiables pero erróneas.

En el caso del coeficiente Kappa, este permite evaluar la concordancia que existe entre las etiquetas reales de cada serie de tiempo y las predicciones realizadas por el clasificador. La ventaja de Kappa es que provee un estadístico que es capaz de cuantificar la calidad de un clasificador. La ecuación que define a Kappa es:

$$k = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$

Donde $Pr(a)$ equivale al acuerdo relativo entre los observadores, y $Pr(e)$ es la probabilidad

hipotética de acuerdo casual, y calculada como:

$$Pr(e) = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

Por último, La curva ROC es una representación gráfica de la sensibilidad frente a la especificidad. Donde la sensibilidad es la relación entre las observaciones positivas pronosticadas correctamente y todas las observaciones en la clase real, mientras que especificidad es la relación entre las observaciones negativas pronosticadas correctamente y todas las observaciones que no son de la clase real.

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

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

El área bajo la curva ROC (AUC) representa el grado o la medida de la separabilidad del modelo, es decir, indica cuánto modelo es capaz de distinguir entre clases. En el caso multoclases, se trazan las curvas ROC para las todas las clase, utilizando la metodología uno contra todos. Posteriormente la AUC resultante será el promedio de todas las AUCs trazadas. Dicho promedio puede ser del tipo macro o micro. Un macro-promedio calcula la métrica de forma independiente para cada clase y luego toma el promedio (por lo tanto, trata a todas las clases por igual), mientras que un micro-promedio agrega las contribuciones de todas las clases para calcular la métrica promedio, es decir, resume los verdaderos positivos individuales, falsos positivos y falsos negativos para diferentes conjuntos y los aplica para obtener las estadísticas.

Además de las métricas anteriores, dado que generalmente los algoritmos de aprendizaje profundo se adaptan bien al desbalance de los datos, y que se está utilizando una validación cruzada se optó por no realizar un preprocessamiento que hiciera un balance de clases, aplicando técnicas como Smote, undersampling, o oversampling. Es por esto que es importante considerar las métricas de análisis que contemplen un desbalance de clases, tomando en cuenta que la AUC favorece las observaciones positivas sobre las negativas. De esta forma, se opta por el uso de la métrica Precisión, Recall y F1-score.C.

En el caso de la precisión, esta mide la relación entre las observaciones positivas pronosticadas correctamente y el total de observaciones positivas pronosticadas.

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

Una alta precisión se relaciona con una baja tasa de falsos positivos

Por otro lado, Recall es la misma sensibilidad.

Y F1-score es el promedio ponderado de Precisión y Recall. Por lo tanto, esta

puntuación tiene en cuenta tanto los falsos positivos como los falsos negativos.

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

Esta medida suele ser más útil que el accuracy, especialmente si tiene una distribución de clases desigual.

Cabe destacar que en el contexto de múltiples clases es preferible un micro-promedio de todas las métricas mencionadas anteriormente, cuando hay presencia de un desbalance de clases. Sin embargo se debe tener en cuenta que el micro-promedio de F1-score, Recall, Precisión y Accuracy dan los mismo valores entre sí, por lo cual en el caso de estas métricas se considerará el macro-promedio para la evaluación de los modelos.

4.3.2 Selección de los modelos

Ya que se cuentan con varios conjuntos de datos de series de tiempo, provenientes de distintas problemáticas, y se implementan varios modelos de aprendizaje, la pregunta es: ¿Alguno de los modelos de aprendizaje es preferible a los demás sobre los conjuntos de datos proporcionados?

Es aquí donde se generan dos hipótesis:

- H_0 : No hay diferencias significativas entre los modelos para los conjuntos de datos considerados.
- H_1 : Existen diferencias significativas entre los modelos para los conjuntos de datos considerados.

Para determinar cuál hipótesis es la correcta, es necesario llevar a cabo un test de hipótesis que analice todas las comparaciones de los modelos buscando encontrar si efectivamente todos los algoritmos tienen el mismo desempeño o, en contraste, algunos de ellos tienen un comportamiento significativamente diferente.

Para ello se aplica el test de Friedman con la extensión de Iman y Davenport que es probablemente el test de hipótesis general más popular, y suele ser una buena opción cuando se comparan más de cinco algoritmos diferentes (Calvo & Santafé, 2016).

El test de Friedman (Análisis bidireccional de varianza de Friedman por rangos) es un test no paramétrico que tiene como objetivo determinar si es posible concluir a partir de una muestra de resultados que existe una diferencia significativa entre los modelos. El primer paso para calcular la estadística del test es convertir los resultados originales a rangos. Por lo tanto, clasifica los algoritmos para cada problema por separado, el mejor algoritmo debe tener el rango 1, el segundo mejor rango 2. En caso de empate, se calculan los rangos promedio. Sea r_i^j el rango j de k modelos en la i de n conjuntos de datos. El test de Friedman compara los rangos

promedio de los algoritmos, $R_j = \frac{1}{n} \sum_i r_i^j$. Bajo la hipótesis nula, que establece que todos los algoritmos son equivalentes y que sus rangos R_j deben ser iguales, la estadística de Friedman es:

$$X_F^2 = \frac{12n}{k(k+1)} \left[\sum j R_j^2 - \frac{k(k+1)^2}{4} \right]$$

Es distribuido de acuerdo a X_F^2 con $k - 1$ grados de libertad, cuando n y k son lo suficientemente grandes (como regla general, $n > 10$ y $k > 5$)

El test de Iman y Davenport es una prueba no paramétrica, derivada de la prueba de Friedman, menos conservadora que la estadística de Friedman:

$$F_F = \frac{(n-1)X_F^2}{n(K-1) - X_F^2}$$

que se distribuye de acuerdo con la distribución F con $k - 1$ y $(k - 1)(n - 1)$ grados de libertad.

En el caso de que el p-valor resultante del test de Friedman con la extensión Iman y Davenport confirme la hipótesis H_1 , denotando que hay al menos un algoritmo que funciona de manera diferente al resto, es necesario proceder con un análisis por pares para comparaciones múltiples entre los modelos a través de un test post-hoc de los resultados (García et al., 2010). En el caso de los test post-hoc, existen varias alternativas para realizar este análisis cuando se realiza una comparación de todos los pares. Para esto se sigue con la línea no paramétrica del test post-hoc de Friedman.

$$z = (R_i - R_j) / \sqrt{\frac{k(k+1)}{6n}}$$

donde R_i y R_j corresponden a los promedios de los rankings de la comparación de los algoritmos con el método de Friedman.

Cuando se considera un p-valor en una comparación múltiple, refleja el error de probabilidad de una cierta comparación, pero no tiene en cuenta las comparaciones restantes que pertenecen al conjunto. Si uno está comparando k modelos y en cada comparación el nivel de significación es α , entonces en una comparación única la probabilidad de no cometer un error de Tipo I (rechazar la hipótesis nula cuando es verdadera) es $(1 - \alpha)$, entonces la probabilidad de no cometer un error de Tipo I en la comparación es $k - 1$ es $1 - (1 - \alpha)^{(k-1)}$. Luego, la probabilidad de cometer uno o más errores de Tipo I es $1 - (1 - \alpha)^{(k-1)}$. Por ejemplo, si $\alpha = 0,05$ y $k = 10$ esto es 0,37, un valor bastante alto. Una forma de resolver este problema es tener los p-valores corregidos (APV), los cuales tienen en cuenta que se están llevando a cabo múltiples pruebas. Un APV puede compararse directamente con cualquier nivel de significación α elegido (García et al., 2010).

El test de Friedman con el APV o corrección de Nemenyi es la alternativa más simple y ampliamente utilizada, pero muy conservadora (Calvo & Santafé, 2016). Su procedimiento

calcula el ajuste del valor de α en un solo paso dividiéndolo por el número de comparaciones consideradas (Settouti et al., 2016). Nemenyi $APV_i : \min\{v; 1\}$, donde $v = k(k - 1)pi/2$. Como resultado se obtienen los p-valores de las diferencias entre cada uno de los modelos y un valor de diferencia crítica que representa el límite sobre el cual se establece que un modelo difiere mucho del otro. La ventaja que presenta el test de Nemenyi es que tiene asociado un gráfico de las diferencias críticas, que representa los resultados de la comparación de los modelos.

Alternativamente, el que tiene la mayor potencia con menor costo computacional es el test post-hoc de Friedman con la corrección de Shaffer (Calvo & Santafé, 2016). La corrección de Shaffer dadas los p-valores obtenidos del test de Friedman para todas las comparaciones por pares, bajo el método de reducción, en la etapa j descarta H_i si p_{it_i} , donde t_i es el número máximo de hipótesis lo cual puede ser cierto dado que cualquier hipótesis $(i, \dots, 1)$ es falsa. (Settouti et al., 2016)

Esto da como resultado los p-valores del test de Friedman con las correcciones de Shaffer de todas las comparaciones entre los pares de modelos, lo que permite bajo cierto umbral encontrar si existen diferencias significativas entre los modelos.

Con estos estadísticos es posible establecer aquellos modelos que efectivamente difieren entre sí, a fin de ir descartando aquellos que son similares. Para la elección final del modelo se tienen en cuenta la evaluación general de cada una de las métricas mencionadas anteriormente, y otras variables como el principio de parsimonia, a fin de elegir el mejor modelo.

4.3.3 Comparación con otros modelos

Modelos considerados

Una vez encontrado el mejor modelo, se establece una comparación con otros modelos de TSC de diversa índole presentes en la literatura. Para ello, se tienen en cuenta diversos papers en los que se han experimentado modelos con conjuntos de datos similares a los que se han considerado para el presente estudio.

La primera comparación se establece con otros modelos pares con enfoque de aprendizaje profundo:

Modelos de aprendizaje profundo

- **MCNN:** CNN multiescala cuya arquitectura de MCNN es muy similar a un modelo tradicional de CNN, con dos convoluciones seguidas de una capa FC y una capa final de softmax, entrenada con subsecuencias extraídas a través del método Window Slicing (WS), en lugar de las series de tiempo de entrada sin procesar (Cui et al., 2016).

- **FCN:** Fully convolutional network, modelo basado en redes neuronales convolucionales, en

el que no se aplica ninguna capa de agrupación local (Wang et al., 2016).

- **ResNet:** Residual Networks, modelo basado en redes neuronales convolucionales, en el que se añade conexiones residuales de atajo entre capas convolucionales consecutivas (Wang et al., 2016).
- **MLP:** Multi Layer Perceptron, forma más tradicional de deep learning, que contempla 4 capas en total, donde cada una está completamente conectada a la salida de su capa anterior, para luego pasar a la capa final softmax para la clasificación (Wang et al., 2016).
- **t-LeNet:** Time-LeNet, modelo inspirado por el gran rendimiento de la arquitectura de LeNet para la tarea de reconocimiento de documentos. Este modelo puede considerarse como una CNN tradicional con dos convoluciones seguidas de una capa FC y un clasificador de softmax final (Fawaz et al., 2019).
- **MCDCCNN:** CNN profunda multicanal donde las convoluciones se aplican de forma independiente en paralelo en cada dimensión de la entrada (Fawaz et al., 2019).
- **Time-CNN:** Modelo convolucional que hace uso del error cuadrático medio (MSE) en lugar de la función de pérdida de entropía cruzada categórica tradicional, e implementa al final una capa FC tradicional con sigmoide como la función de activación, que no garantiza una suma de probabilidades igual a 1 (Fawaz et al., 2019).
- **TWIESN:** Red de Eco Invariante de Time Warping es una arquitectura recurrente no convolucional originalmente propuesta para el pronóstico de series de tiempo, que usa directamente la serie de tiempo de entrada sin procesar y predice una distribución de probabilidad sobre las variables de clase, a través de la proyección de cada elemento de la serie de tiempo en un espacio dimensional superior (Fawaz et al., 2019).
- **Encoder:** CNN híbrido profundo cuya arquitectura está inspirada en el modelo FCN, con una diferencia principal donde la capa GAP se reemplaza por una capa de atención (Fawaz et al., 2019).

Posteriormente se establece otra comparación con modelos de la literatura con diversos enfoques que presentan un rendimiento destacado.

Modelos basados en distancia

- **DTW_F:** Dinamyc Tima Warping, tiene una medida de distancia de tiempo como metrica del algoritmo de K vecinos más cercanos (KNN) (Lines et al., 2018).

Modelos basados en características

- **BOSS:** Bag of SFA Symbols, modelo que extrae subestructuras de características deslizando ventanas y usando un método de reducción de ruido (Schäfer, 2015a).
- **TSF:** TS of Features framework, modelo que emplea una combinación de la ganancia de entropía y una medida de distancia para la extracción de características para luego hacer la clasificación con Random forest (Deng et al., 2013b).
- **TSBF:** TS Bag of Features framework, modelo similar a TSF que emplea múltiples subsecuencias seleccionadas de ubicaciones y longitudes aleatorias que se dividen en intervalos más cortos para capturar la información local (Deng et al., 2013c).
- **LPS:** Clasificador basado en intervalos en el que las subseries se convierten en atributos para luego crear un modelo de regresión interno para detectar correlaciones entre subseries (Lines et al., 2018).

Modelos basados en conjuntos

- **COTE:** Colectivo de conjuntos basados en la transformación, modelo que combina distintos clasificadores construidos en los dominios de tiempo, frecuencia y de la transformación Shapelet (Bagnall et al., 2015).
- **HIVE-COTE:** Colectivo de conjuntos basados en la transformación, modelo similar a COTE que propone una estructura jerárquica con votación probabilística, definiendo e incluyendo más clasificadores en espacios de características existentes y agregando más módulos para representar dos dominios de transformación adicionales. (Lines et al., 2018).
- **ST-HESCA:** Conjunto heterogéneo de algoritmos de clasificación estándar, combinado con conjuntos que realizan transformación de Shapelet (Lines et al., 2018).
- **EE:** Elastic Ensemble, clasificador que combina diversos clasificadores que utilizan medidas de distancia elástica (Lines & Bagnall, 2015).

Comparación estadística

Para esto, se realiza una pregunta similar a la que se establece en el procedimiento de la selección del mejor modelo.

¿Alguno de los modelos de aprendizaje presenta un rendimiento mejor que los demás sobre los conjuntos de datos proporcionados?

Las hipótesis también son similares, por lo que se procede a aplicar el Test de Friedman con la extensión de Iman y Davenport para detectar si existen diferencias significativas entre los modelos.

Si se encuentran diferencias significativas, se procede a aplicar el test post-hoc de Friedman con

la corrección de Shaffer para las comparaciones múltiples por pares. A manera de apoyo, se incluye también el test de Nemenyi de las comparaciones por pares para todos los modelos, de manera de obtener el gráfico de las diferencias críticas, aprovechado que esta una herramienta visual fácil de entender.

CAPÍTULO 5. RESULTADOS

En el presente capítulo se presentan los resultados obtenidos con los modelos de arquitecturas profundas en su entrenamiento con todos los conjuntos de datos considerados, bajo las métricas expuestas en el capítulo anterior. Además se presenta los resultados del proceso de la selección del mejor modelo implementado a través de un test de hipótesis para ver si existen diferencias significativas entre los modelos, y en el caso de que sea así, se presentan test post-hoc para la selección final.

Con el mejor modelo seleccionado, se presentan las comparaciones con algunos de los modelos implementados en la literatura, para establecer si el modelo implementado presenta un rendimiento de vanguardia o no.

5.1 SELECCIÓN DEL MEJOR MODELO IMPLEMENTADO

Con las 10 implementaciones de los modelos antes considerados, se llevó a cabo pruebas con los 44 conjuntos de datos de la UCR, utilizando validación cruzada de 5-fold. Esto da como total 2200 pruebas, que fueron llevadas a cabo en 2 laptops y la herramienta *Google Colab*, servicio en la nube gratuito basado en Jupyter Notebooks que dispone del uso de una GPU k80. En total el tiempo de procesamiento fue de aproximadamente 120 días, incluyendo las primeras pruebas y la grilla de parámetros inicial, generándose así alrededor de 50 Gb en datos. Los resultados generales se encumbran a continuación.

En primer lugar se muestran gráficos de radar que ayudan a visualizar los rendimientos de cada implementación con cada una de las métricas para establecer conclusiones a priori del rendimiento de cada uno de los modelos.

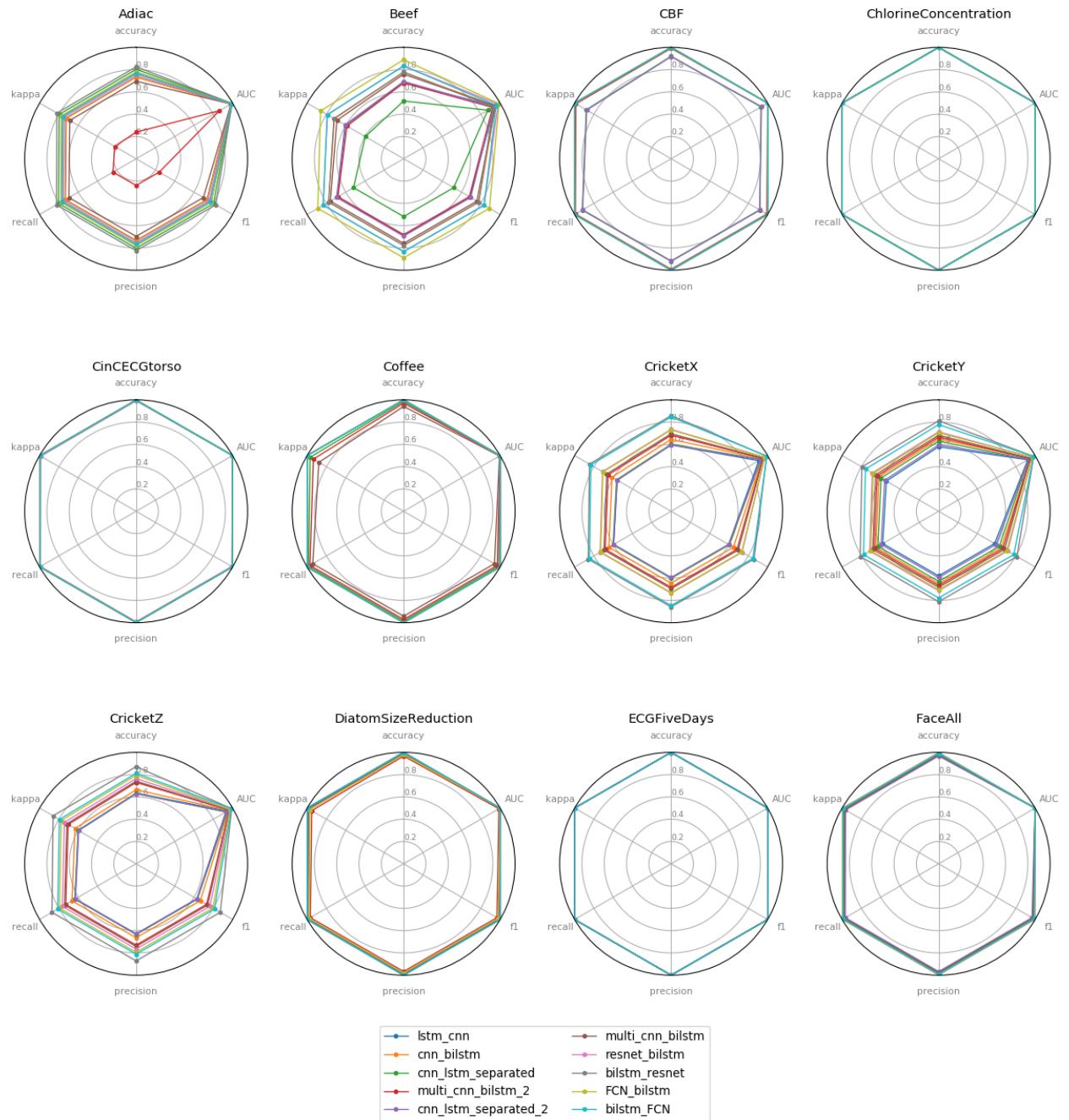


Figura 5.1: Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde Adiac, hasta FaceAll .

Fuente: Elaboración propia, (2019).

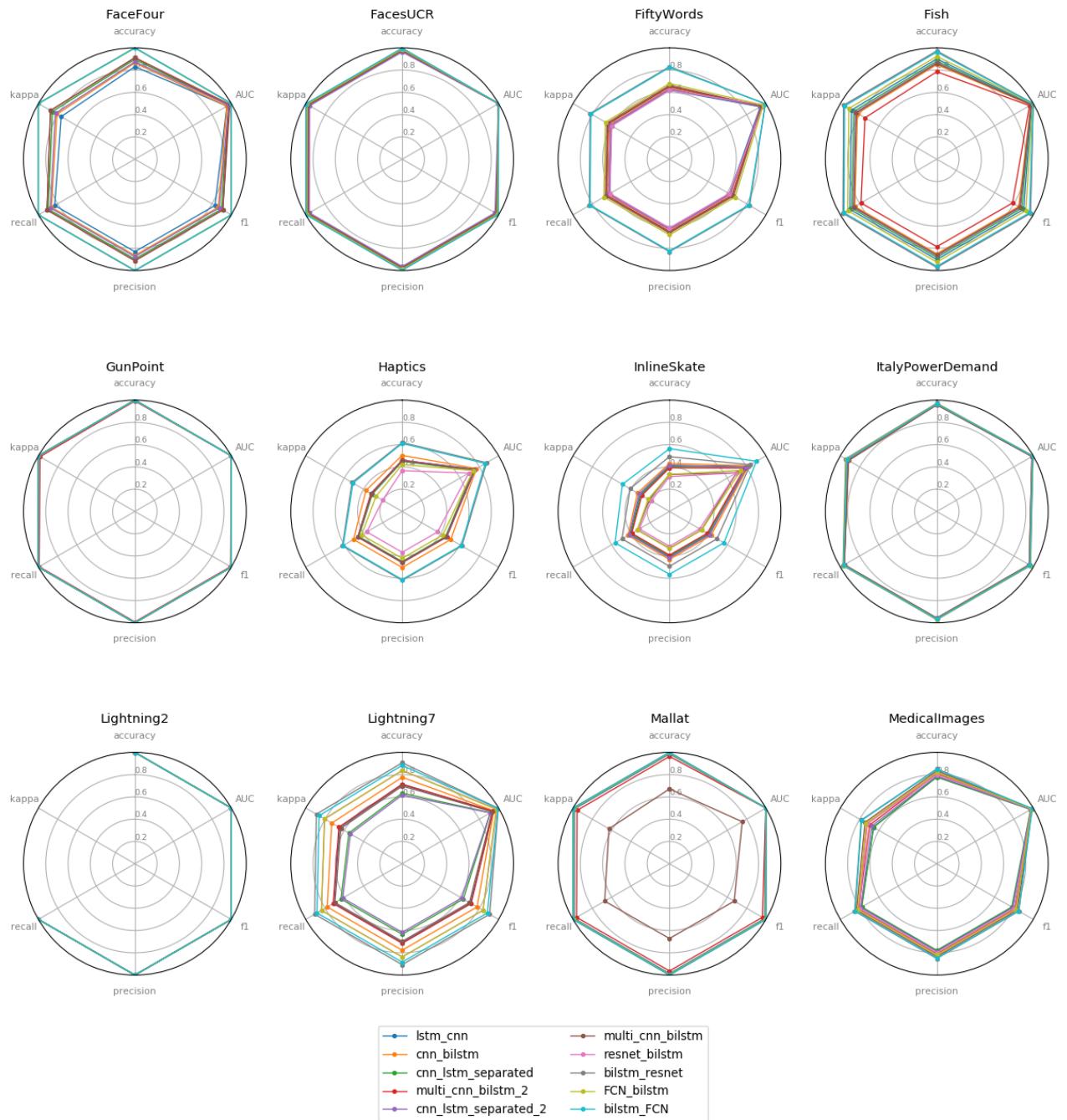


Figura 5.2: Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde FaceFour, hasta MedicalImages.

Fuente: Elaboración propia, (2019).

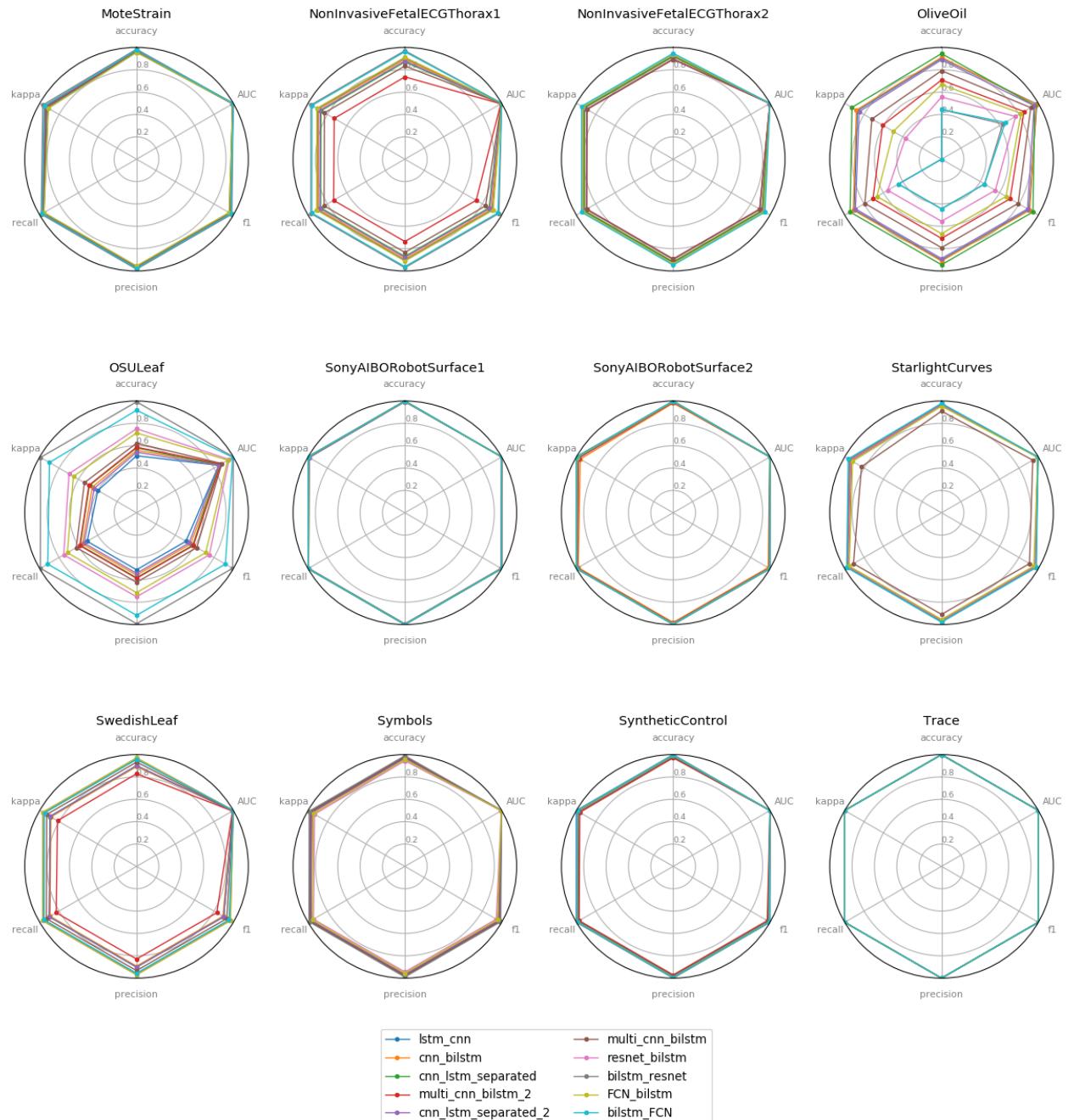


Figura 5.3: Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, MoteStrain hasta Trace.
Fuente: Elaboración propia, (2019).

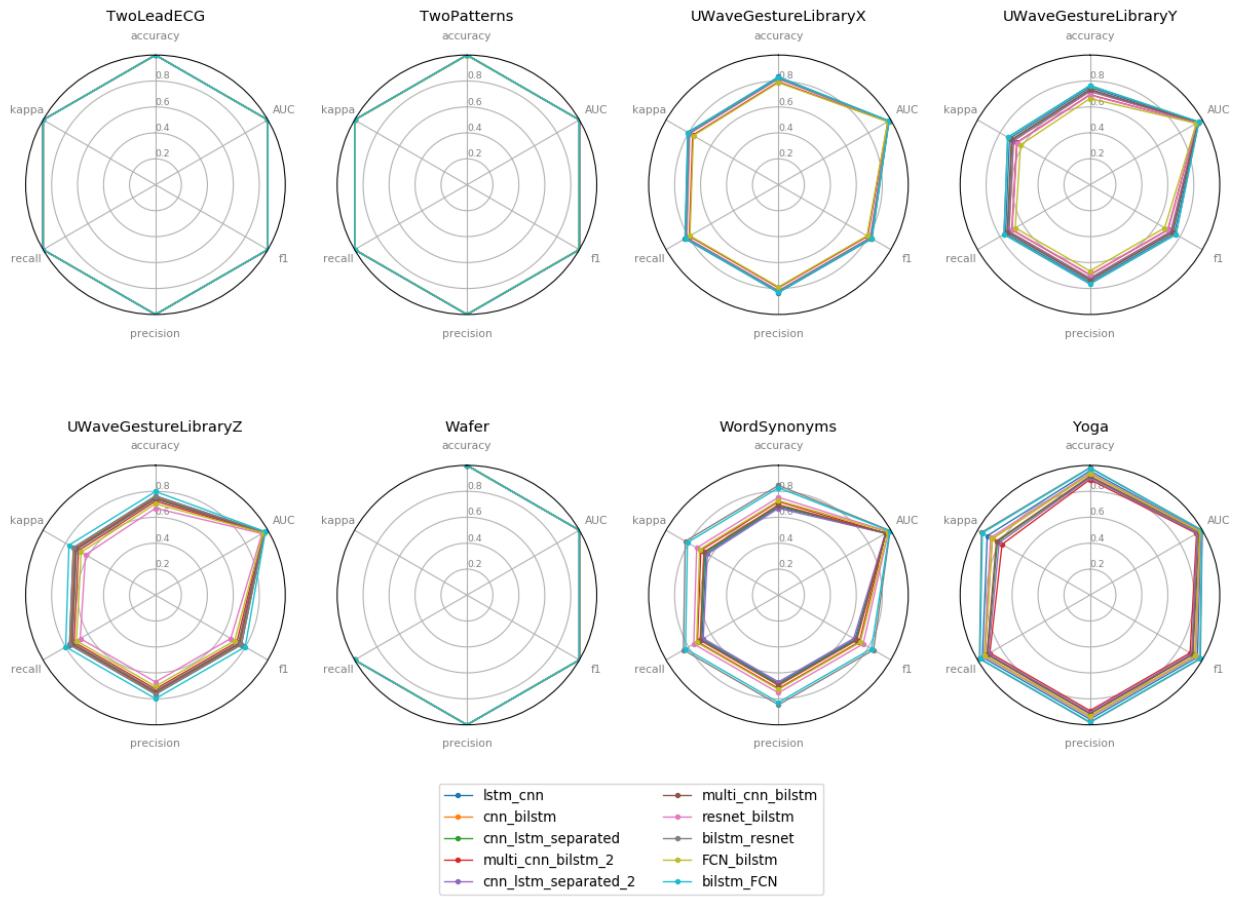


Figura 5.4: Gráfico radar de las métricas para cada uno de los modelos en cada uno de los conjuntos de datos, desde TwoLeadECG, hasta Yoga.

Fuente: Elaboración propia, (2019).

Como es posible apreciar, la lista de los conjuntos de datos en los que todos los modelos presentan un muy bajo rendimiento para todas las métricas consideradas (menor a 0,6), son Haptics e InlineSkate.

Bajo esta misma linea, otros conjuntos de datos en los que la mayoría de los modelos presentan bajo rendimiento (tener una medida con un mínimo de 0,8 en cada una de las métricas), son: Adiac - Beef - CricketX - CricketY - CricketZ - FiftyWords - Lightning7 - OliveOil - OsuLeaf - UWaveGestureLibraryX - UWaveGestureLibraryY - UWaveGestureLibraryZ - WordSynonyms

Estos conjuntos de datos son interesantes ya que en su rendimiento bajo con todos los modelos, pueden constituir un punto de discusión.

Se observa a su vez a priori, que todos los modelos tienen un comportamiento similar bajo todas las métricas, salvo por algunos conjuntos de datos en los que se observan diferencias apreciables entre algunos modelos, especialmente en los que se mencionan anteriormente. Bajo esta mirada, es posible observar que el modelo multi_cnn_bilstm2 en general es el que presenta un rendimiento más bajo.

A continuación se presentan los resultados generales para todos los modelos y los conjuntos de datos bajo la métrica del coeficiente de kappa. Así mismo, los resultados bajo la métrica del error se pueden encontrar en el Apéndice A.

Para la obtención de conocimiento a partir de los resultados presentados se establecen dos parámetros. El primero de ellos, "Ganados", que cuenta en cantidad las veces en que el modelo presenta el mejor rendimiento en cierto conjunto de datos, en comparación con los otros modelos a pesar de que haya un empate en rendimiento. El segundo corresponde al promedio de ranking de Friedman.

Tabla 5.1: Coeficientes Kappa promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.

Conjuntos de da- tos	lstm_cnn	cnn_ bilstm	cnn_lstm_ separa- ted	multi_cnn_ bilstm2	cnn_lstm_ separa- ted2	multi_cnn_ bilstm	resnet_ bilstm	bilstm_ resnet	FCN_ bilstm	bilstm_ FCN
Adiac	0.73	0.72	0.8	0.22	0.77	0.69	0.75	0.82	0.77	0.76
Beef	0.6	0.72	0.4	0.59	0.61	0.69	0.79	0.72	0.86	0.79
CBF	1.0	1.0	0.88	0.99	0.87	1.0	1.0	1.0	1.0	1.0
ChlorineConcn	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CinCECGtorso	1.0	1.0	1.0	1.0	0.99	1.0	0.99	1.0	1.0	1.0
Coffee	1.0	1.0	0.97	0.94	1.0	0.88	1.0	1.0	1.0	1.0
CricketX	0.56	0.61	0.57	0.65	0.56	0.67	0.71	0.85	0.71	0.83
CricketY	0.54	0.62	0.6	0.64	0.56	0.65	0.69	0.79	0.68	0.76
CricketZ	0.6	0.63	0.6	0.7	0.59	0.72	0.74	0.86	0.78	0.79
DiatomSizeR	1.0	0.99	0.99	0.95	0.99	0.99	1.0	1.0	0.97	1.0
ECGFiveDays	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FaceAll	0.97	0.98	0.97	0.98	0.96	0.98	1.0	1.0	0.99	0.99
FaceFour	0.76	0.81	0.85	0.87	0.82	0.87	1.0	1.0	1.0	1.0
FacesUCR	0.97	0.97	0.97	0.97	0.96	0.98	0.99	1.0	0.99	1.0
FiftyWords	0.6	0.62	0.63	0.63	0.61	0.65	0.6	0.82	0.66	0.81
Fish	0.88	0.82	0.86	0.74	0.84	0.83	0.97	0.97	0.91	0.96
GunPoint	1.0	1.0	1.0	0.98	1.0	1.0	1.0	1.0	1.0	1.0
Haptics	0.32	0.38	0.31	0.32	0.32	0.32	0.2	0.52	0.27	0.51
InlineSkate	0.3	0.33	0.3	0.29	0.32	0.21	0.19	0.41	0.22	0.49
ItalyPowerDem	0.92	0.91	0.92	0.92	0.92	0.92	0.93	0.95	0.94	0.93
Lightning2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Lightning7	0.65	0.73	0.55	0.66	0.54	0.64	0.81	0.89	0.8	0.86

Mallat	0.98	1.0	1.0	0.96	0.99	0.63	1.0	1.0	0.99	1.0
MedicallImages	0.66	0.72	0.66	0.69	0.67	0.74	0.72	0.78	0.73	0.78
MoteStrain	0.91	0.93	0.93	0.95	0.93	0.93	0.95	0.97	0.91	0.95
NIInvFetalECGThx1	0.9	0.89	0.87	0.73	0.88	0.83	0.95	0.97	0.91	0.96
NIInvFetalECGThx2	0.92	0.91	0.92	0.89	0.9	0.89	0.93	0.95	0.93	0.95
OliveOil	0.87	0.88	0.93	0.61	0.85	0.72	0.37	0.0	0.5	0.0
OSULeaf	0.4	0.46	0.49	0.49	0.44	0.54	0.7	0.99	0.65	0.9
SonyAIBORobotS1	1.0	0.99	0.99	1.0	0.99	1.0	1.0	1.0	1.0	1.0
SonyAIBORobotS2	0.99	0.96	0.99	0.98	0.98	0.99	1.0	1.0	0.99	1.0
StarlightCurves	0.95	0.94	0.93	0.94	0.93	0.83	0.94	0.95	0.92	0.96
SwedishLeaf	0.89	0.88	0.89	0.82	0.89	0.93	0.97	0.96	0.97	0.95
Symbols	0.98	0.96	0.97	0.96	0.96	0.98	0.93	0.99	0.95	0.98
SyntheticControl	1.0	0.99	0.99	0.96	0.99	0.97	0.99	0.99	0.99	0.99
Trace	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TwoLeadECG	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TwoPatterns	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
UWaveGestureLibX	0.81	0.79	0.79	0.76	0.8	0.8	0.79	0.79	0.76	0.8
UWaveGestureLibY	0.72	0.68	0.69	0.65	0.68	0.7	0.65	0.73	0.62	0.73
UWaveGestureLibZ	0.73	0.72	0.7	0.68	0.71	0.71	0.62	0.73	0.66	0.77
Wafer	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WordSynonyms	0.64	0.66	0.66	0.69	0.63	0.66	0.73	0.83	0.7	0.81
Yoga	0.91	0.87	0.82	0.78	0.81	0.83	0.88	0.97	0.87	0.96
Ganados	15	12	11	9	9	11	18	36	15	23
Ranking Promedio	5.886 (5)	6.090 (7)	6.465 (8)	7.125 (10)	7.022 (9)	6.079 (6)	4.909 (3)	2.920 (1)	5.227 (4)	3.272 (2)

Fuente: Elaboración propia, (2019).

De la Tabla 5.1, en términos de la contabilidad de los conjuntos de datos totales donde cada modelo tiene un mejor rendimiento bajo la métrica del coeficiente de Kappa, definiendo el grado de concordancia entre las muestras, es posible dirimir con una gran diferencia respecto a los otros modelos, que el mejor es el de bilstm_resnet con 36 en total, teniendo en cuenta que hay ciertos conjuntos de datos donde existe empate entre los resultados. A continuación le siguen los modelos bilstm_FCN y resnet_bilstm con 23 y 18 respectivamente. Cabe destacar que estos tres modelos están basados en arquitecturas encontradas en la literatura, a los que se le agrega una capa recurrente bilstm ya sea al principio o al final de las otras capas convolucionales. Bajo estos mismos términos, es posible establecer cierta paridad entre los modelos multi_cnn_bilstm2, cnn_lstm_separated y cnn_lstm_separated2, considerando además el promedio total de la métrica.

Por otro lado los algoritmos que presentan un peor rendimiento en términos de cantidad de ganados y del promedio total de la métrica son multi_cnn_bilstm2 y cnn_lstm_separated tal como se venía adelantando con la información que se pudo obtener a través de los gráficos de radar.

Un aspecto importante a considerar para el análisis de los resultados es el dominio del problema que abarca el conjunto de datos. Es entonces que dados los dominios de los conjuntos de datos, se establece un promedio general de cada una de las métricas consideradas, cuyos resultados se presentan una serie de tablas a continuación, siendo cada tabla correspondiente a un dominio determinado.

Tabla 5.2: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo ECG (5). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
Istm_cnn	0.966 (0.04)	0.999 (0.0)	0.966 (0.04)	0.967 (0.04)	0.967 (0.04)	0.965 (0.04)	0.131 (0.16)
cnn_bilstm	0.962 (0.05)	0.999 (0.0)	0.962 (0.05)	0.965 (0.04)	0.962 (0.05)	0.961 (0.05)	0.168 (0.21)
cnn_lstm_separated	0.958 (0.05)	0.999 (0.0)	0.957 (0.05)	0.959 (0.05)	0.958 (0.05)	0.956 (0.05)	0.161 (0.19)
multi_cnn_bilstm2	0.925 (0.1)	0.997 (0.0)	0.921 (0.11)	0.936 (0.08)	0.924 (0.1)	0.923 (0.11)	0.384 (0.48)
cnn_lstm_separated2	0.954 (0.05)	0.999 (0.0)	0.954 (0.06)	0.955 (0.05)	0.955 (0.05)	0.953 (0.06)	0.167 (0.2)
multi_cnn_bilstm	0.945 (0.07)	0.999 (0.0)	0.943 (0.07)	0.95 (0.06)	0.945 (0.07)	0.943 (0.07)	0.275 (0.34)
resnet_bilstm	0.982 (0.03)	1.0 (0.0)	0.981 (0.03)	0.982 (0.03)	0.981 (0.03)	0.98 (0.03)	0.065 (0.09)
bilstm_resnet	0.992 (0.01)	1.0 (0.0)	0.992 (0.01)	0.992 (0.01)	0.992 (0.01)	0.992 (0.01)	0.025 (0.04)
FCN_bilstm	0.961 (0.04)	0.999 (0.0)	0.961 (0.04)	0.963 (0.04)	0.961 (0.04)	0.96 (0.04)	0.151 (0.14)
bilstm_FCN	0.981 (0.02)	1.0 (0.0)	0.981 (0.02)	0.982 (0.02)	0.981 (0.02)	0.981 (0.02)	0.054 (0.07)

Fuente: Elaboración propia, (2019).

Del dominio del tipo de electrocardiogramas (ECG), Tabla 5.2, que engloba 5 conjuntos de datos en total, se destaca que el mejor rendimiento lo posee el modelo bilstm_resnet, seguido muy de cerca por los otros modelos. En general dentro de este dominio, la mayoría de los modelos se ajusta bien, logrando un buen rendimiento.

Del dominio del tipo de imágenes (IMAGES) con 13 conjuntos de datos, Tabla 5.3, se destaca que el mejor rendimiento en todas las métricas lo posee el modelo bilstm_resnet, seguido

Tabla 5.3: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo IMAGE (13). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.835 (0.15)	0.971 (0.04)	0.8 (0.19)	0.815 (0.18)	0.807 (0.19)	0.8 (0.18)	0.64 (0.56)
cnn_bilstm	0.838 (0.14)	0.972 (0.03)	0.805 (0.18)	0.818 (0.16)	0.809 (0.17)	0.805 (0.16)	0.654 (0.56)
cnn_lstm_separated	0.847 (0.13)	0.971 (0.04)	0.812 (0.18)	0.827 (0.16)	0.815 (0.17)	0.812 (0.15)	1.061 (0.99)
multi_cnn_bilstm2	0.794 (0.2)	0.96 (0.04)	0.76 (0.22)	0.79 (0.21)	0.761 (0.21)	0.756 (0.21)	1.349 (1.39)
cnn_lstm_separated2	0.834 (0.14)	0.97 (0.04)	0.799 (0.19)	0.816 (0.17)	0.803 (0.18)	0.798 (0.16)	0.878 (0.8)
multi_cnn_bilstm	0.851 (0.13)	0.976 (0.03)	0.819 (0.17)	0.842 (0.15)	0.821 (0.16)	0.82 (0.15)	0.981 (1.03)
resnet_bilstm	0.884 (0.12)	0.989 (0.01)	0.86 (0.17)	0.867 (0.16)	0.864 (0.16)	0.864 (0.14)	0.459 (0.5)
bilstm_resnet	0.942 (0.07)	0.997 (0.0)	0.927 (0.1)	0.937 (0.08)	0.926 (0.1)	0.932 (0.08)	0.206 (0.25)
FCN_bilstm	0.882 (0.12)	0.987 (0.02)	0.857 (0.15)	0.871 (0.14)	0.86 (0.15)	0.859 (0.13)	0.512 (0.53)
bilstm_FCN	0.922 (0.08)	0.996 (0.0)	0.905 (0.11)	0.915 (0.09)	0.906 (0.1)	0.909 (0.09)	0.262 (0.29)

Fuente: Elaboración propia, (2019).

por el modelo bilstm_FCN con un rendimiento no significativamente menor. En general dentro de este dominio se evidencian mayores diferencias entre los resultados que obtienen los modelos, manejando un rango entre 0,75 y 0,93 lo que exalta el rendimiento de los modelos anteriormente mencionados.

Tabla 5.4: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo MOTION (9). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.669 (0.18)	0.919 (0.08)	0.664 (0.18)	0.676 (0.17)	0.669 (0.17)	0.62 (0.21)	0.972 (0.51)
cnn_bilstm	0.688 (0.16)	0.928 (0.07)	0.682 (0.16)	0.69 (0.16)	0.689 (0.16)	0.641 (0.19)	0.915 (0.45)
cnn_lstm_separated	0.667 (0.17)	0.913 (0.08)	0.662 (0.17)	0.67 (0.17)	0.669 (0.17)	0.617 (0.21)	1.828 (0.92)
multi_cnn_bilstm2	0.682 (0.17)	0.919 (0.08)	0.676 (0.17)	0.688 (0.16)	0.68 (0.17)	0.632 (0.2)	2.134 (1.3)
cnn_lstm_separated2	0.664 (0.17)	0.914 (0.08)	0.657 (0.17)	0.67 (0.17)	0.667 (0.17)	0.615 (0.21)	1.481 (0.77)
multi_cnn_bilstm	0.69 (0.18)	0.912 (0.1)	0.681 (0.19)	0.689 (0.19)	0.689 (0.19)	0.642 (0.23)	3.144 (3.84)
resnet_bilstm	0.674 (0.2)	0.907 (0.12)	0.668 (0.21)	0.683 (0.2)	0.673 (0.2)	0.621 (0.25)	1.075 (0.65)
bilstm_resnet	0.778 (0.14)	0.955 (0.05)	0.775 (0.14)	0.785 (0.13)	0.78 (0.14)	0.743 (0.17)	0.645 (0.38)
FCN_bilstm	0.682 (0.19)	0.918 (0.1)	0.678 (0.19)	0.691 (0.18)	0.682 (0.19)	0.632 (0.23)	0.953 (0.48)
bilstm_FCN	0.778 (0.12)	0.962 (0.04)	0.773 (0.13)	0.78 (0.12)	0.777 (0.12)	0.743 (0.15)	0.601 (0.31)

Fuente: Elaboración propia, (2019).

En general en el dominio del movimiento (MOTION) que reúne 9 conjuntos de datos, Tabla 5.4, todos los modelos presentan un rendimiento medio, en el rango de 0,667 a 0,778, lo que revela las dificultades para el proceso de aprendizaje dadas las características de este tipo de dominio.

Esta vez existe gran paridad entre los modelos bilstm_resnet y bilstm_FCN, destacándose este último en términos del error promedio.

En el dominio de los sensores (SENSOR) con 9 conjuntos de datos, Tabla 5.5, todos los modelos presentan un buen rendimiento, superior a 0,83 en todas las métricas, reflejando que el aprendizaje se adapta bien a este tipo de modelos. El más destacable es nuevamente el modelo bilstm_resnet, seguido muy de cerca por el modelo bilstm_FCN.

De la Tabla 5.6, bajo la mirada del dominio simulación (SIMULATED) que engloba 5 conjuntos de datos, Tabla 5.6, se destaca el excelente rendimiento de todos los modelos en

Tabla 5.5: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SENSOR (9). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.954 (0.09)	0.991 (0.02)	0.951 (0.09)	0.956 (0.08)	0.951 (0.09)	0.917 (0.11)	0.136 (0.26)
cnn_bilstm	0.961 (0.07)	0.992 (0.02)	0.957 (0.07)	0.96 (0.06)	0.957 (0.07)	0.924 (0.08)	0.143 (0.25)
cnn_lstm_separated	0.946 (0.11)	0.988 (0.03)	0.94 (0.12)	0.947 (0.11)	0.939 (0.12)	0.902 (0.15)	0.296 (0.62)
multi_cnn_bilstm2	0.956 (0.09)	0.991 (0.02)	0.953 (0.09)	0.958 (0.08)	0.954 (0.08)	0.919 (0.11)	0.227 (0.43)
cnn_lstm_separated2	0.941 (0.12)	0.987 (0.03)	0.937 (0.13)	0.945 (0.11)	0.939 (0.13)	0.893 (0.16)	0.28 (0.58)
multi_cnn_bilstm	0.947 (0.09)	0.984 (0.03)	0.932 (0.11)	0.934 (0.1)	0.936 (0.1)	0.899 (0.12)	0.53 (0.92)
resnet_bilstm	0.972 (0.05)	0.997 (0.01)	0.97 (0.05)	0.974 (0.04)	0.969 (0.05)	0.947 (0.06)	0.09 (0.16)
bilstm_resnet	0.982 (0.03)	0.999 (0.0)	0.981 (0.03)	0.981 (0.03)	0.981 (0.03)	0.965 (0.04)	0.049 (0.07)
FCN_bilstm	0.968 (0.05)	0.995 (0.01)	0.961 (0.06)	0.971 (0.04)	0.962 (0.06)	0.938 (0.07)	0.102 (0.17)
bilstm_FCN	0.978 (0.04)	0.998 (0.0)	0.977 (0.04)	0.977 (0.04)	0.977 (0.04)	0.959 (0.05)	0.061 (0.1)

Fuente: Elaboración propia, (2019).

Tabla 5.6: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SIMULATED (5). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.997 (0.01)	1.0 (0.0)	0.997 (0.01)	0.997 (0.01)	0.997 (0.01)	0.997 (0.01)	0.01 (0.02)
cnn_bilstm	0.997 (0.0)	1.0 (0.0)	0.997 (0.0)	0.997 (0.0)	0.997 (0.0)	0.996 (0.01)	0.007 (0.01)
cnn_lstm_separated	0.982 (0.03)	0.986 (0.03)	0.975 (0.04)	0.973 (0.05)	0.981 (0.03)	0.973 (0.05)	0.553 (1.09)
multi_cnn_bilstm2	0.985 (0.02)	1.0 (0.0)	0.985 (0.01)	0.986 (0.01)	0.985 (0.01)	0.981 (0.02)	0.057 (0.06)
cnn_lstm_separated2	0.981 (0.03)	0.987 (0.03)	0.975 (0.05)	0.97 (0.05)	0.981 (0.03)	0.972 (0.05)	0.581 (1.15)
multi_cnn_bilstm	0.929 (0.13)	0.951 (0.1)	0.919 (0.15)	0.915 (0.16)	0.932 (0.12)	0.919 (0.15)	2.267 (4.48)
resnet_bilstm	0.999 (0.0)	1.0 (0.0)	0.999 (0.0)	0.999 (0.0)	0.999 (0.0)	0.998 (0.0)	0.003 (0.01)
bilstm_resnet	0.999 (0.0)	1.0 (0.0)	0.999 (0.0)	0.999 (0.0)	0.999 (0.0)	0.999 (0.0)	0.004 (0.01)
FCN_bilstm	0.997 (0.0)	1.0 (0.0)	0.997 (0.0)	0.997 (0.0)	0.997 (0.0)	0.996 (0.0)	0.009 (0.01)
bilstm_FCN	0.999 (0.0)	1.0 (0.0)	0.999 (0.0)	0.999 (0.0)	0.999 (0.0)	0.999 (0.0)	0.002 (0.0)

Fuente: Elaboración propia, (2019).

general considerando todas las métricas evaluadas.

De aquí se destaca el casi perfecto rendimiento que presentan los modelos resnet_bilstm, bilstm_resnet y bilstm_FCN, muy parejos entre sí.

Tabla 5.7: Resultados promedio clasificados para cada modelo por métrica en los conjuntos de datos del tipo SPECTRO (3). Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.863 (0.13)	0.976 (0.02)	0.833 (0.17)	0.833 (0.17)	0.846 (0.16)	0.823 (0.17)	0.337 (0.29)
cnn_bilstm	0.896 (0.09)	0.976 (0.02)	0.877 (0.12)	0.888 (0.11)	0.882 (0.11)	0.866 (0.12)	0.349 (0.26)
cnn_lstm_separated	0.816 (0.21)	0.944 (0.05)	0.788 (0.25)	0.796 (0.24)	0.801 (0.24)	0.764 (0.26)	0.614 (0.56)
multi_cnn_bilstm2	0.786 (0.13)	0.924 (0.06)	0.721 (0.17)	0.725 (0.17)	0.758 (0.15)	0.71 (0.16)	1.024 (0.79)
cnn_lstm_separated2	0.861 (0.13)	0.962 (0.03)	0.832 (0.17)	0.833 (0.17)	0.845 (0.16)	0.819 (0.16)	0.523 (0.44)
multi_cnn_bilstm	0.831 (0.08)	0.955 (0.03)	0.786 (0.11)	0.785 (0.11)	0.814 (0.1)	0.763 (0.08)	0.616 (0.37)
resnet_bilstm	0.796 (0.18)	0.915 (0.11)	0.727 (0.28)	0.72 (0.31)	0.773 (0.22)	0.722 (0.26)	0.532 (0.5)
bilstm_resnet	0.741 (0.23)	0.855 (0.16)	0.645 (0.36)	0.64 (0.38)	0.683 (0.32)	0.574 (0.42)	0.998 (0.92)
FCN_bilstm	0.852 (0.14)	0.936 (0.08)	0.764 (0.26)	0.769 (0.28)	0.79 (0.23)	0.786 (0.21)	0.47 (0.44)
bilstm_FCN	0.759 (0.23)	0.873 (0.15)	0.668 (0.37)	0.654 (0.39)	0.7 (0.32)	0.597 (0.43)	0.646 (0.53)

Fuente: Elaboración propia, (2019).

En lo que se refiere al dominio de los espectros que presenta 3 conjuntos de datos, Tabla 5.7, el rendimiento general de todos los modelos podría clasificarse como medio alto, no sobrepasando el 0,9 para todas las métricas.

Aquí a diferencia de la tendencia reflejada en los otros dominios en los que se evidenciaba cierta superioridad del modelo bilstm_resnet, seguido por bilstm_FCN, ambos presentan bajas en el rendimiento, llegando incluso a tener menor rendimiento que el resto de los modelos. Esta vez quien se destaca es el modelo cnn_bilstm, seguido de cerca por lstm_cnn, quienes a lo largo de todos los dominios presentaron un rendimiento promedio.

Finalmente en la Tabla 5.8, se presentan los promedios totales de cada una de las métricas para todos los modelos en sus pruebas con todos los conjuntos de datos. Si bien el uso de la ponderación de cada una de las métricas introduce otro componente de subjetividad, considerando que los dominios de los conjuntos de datos son distintos, se utiliza a modo de resumen para ayudar en la identificación del mejor modelo general, antes de realizar el proceso de la debida selección apoyado por tests de hipótesis.

Tabla 5.8: Resultados promedio clasificados con todos los conjuntos de datos para cada modelo por métrica. Entre paréntesis su desviación estándar.

	accuracy	AUC	f1	precision	recall	kappa	log_loss
lstm_cnn	0.861 (0.17)	0.971 (0.05)	0.847 (0.19)	0.855 (0.18)	0.851 (0.18)	0.826 (0.2)	0.455 (0.54)
cnn_bilstm	0.869 (0.15)	0.974 (0.05)	0.855 (0.17)	0.863 (0.16)	0.859 (0.17)	0.835 (0.18)	0.453 (0.51)
cnn_lstm_separated	0.856 (0.17)	0.965 (0.06)	0.841 (0.19)	0.849 (0.18)	0.845 (0.19)	0.818 (0.2)	0.871 (1.02)
multi_cnn_bilstm2	0.84 (0.18)	0.964 (0.06)	0.823 (0.2)	0.838 (0.19)	0.828 (0.19)	0.8 (0.21)	1.001 (1.26)
cnn_lstm_separated2	0.851 (0.17)	0.966 (0.05)	0.836 (0.19)	0.845 (0.18)	0.841 (0.18)	0.813 (0.2)	0.751 (0.87)
multi_cnn_bilstm	0.856 (0.16)	0.963 (0.07)	0.837 (0.18)	0.846 (0.17)	0.844 (0.17)	0.817 (0.19)	1.372 (2.63)
resnet_bilstm	0.875 (0.17)	0.971 (0.07)	0.861 (0.2)	0.867 (0.19)	0.866 (0.19)	0.842 (0.21)	0.427 (0.58)
bilstm_resnet	0.913 (0.13)	0.979 (0.06)	0.901 (0.17)	0.906 (0.16)	0.905 (0.15)	0.884 (0.19)	0.28 (0.45)
FCN_bilstm	0.877 (0.16)	0.974 (0.06)	0.861 (0.18)	0.87 (0.18)	0.864 (0.18)	0.844 (0.19)	0.424 (0.51)
bilstm_FCN	0.908 (0.13)	0.982 (0.06)	0.896 (0.16)	0.899 (0.16)	0.899 (0.15)	0.878 (0.19)	0.263 (0.35)

Fuente: Elaboración propia, (2019).

Siguiendo la tónica de la mayoría de los dominios, y de las inferencias anteriormente establecidas, se puede establecer a priori, a pesar de todos los elementos subjetivos que podría tener usar solo la examinación del promedio de cada una de las métricas, que los mejores modelos son bilstm_resnet y bilstm_FCN, rescatando que éste último tiene el promedio de error más bajo, aunque con una diferencia no muy significativa. Por otro lado, los modelos con menor rendimiento en general son multi_cnn_bilstm2, cnn_lstm_separated2, multi_cnn_bilstm y cnn_lstm_separated registrando rendimientos promedio muy pares entre sí, reflejando la similitud de sus arquitecturas.

Otras herramientas útiles para el análisis del rendimiento de cada uno de los modelos son su performance según las características de cada conjunto de datos. De esta manera, se distinguen el ranking promedio según la métrica kappa que obtienen cada uno de los modelos en base a la cantidad de clases, número de instancias que tiene cada conjunto de datos y por la longitud de las series de tiempo. Los resultados se presentan en las tablas 5.9, 5.10 y 5.11

respectivamente, donde los valores más bajos implican un mejor rendimiento.

Tabla 5.9: Promedio de ranking por cantidad de clases del conjunto de datos.

	2	3-5	6-10	11-20	>20
lstm_cnn	5.45	5.14	5.05	8.50	7.20
cnn_bilstm	6.55	5.00	5.68	7.25	7.00
cnn_lstm_separated	6.45	6.36	6.18	7.92	5.60
multi_cnn_bilstm2	6.68	6.36	8.05	6.92	8.00
cnn_lstm_separated2	6.45	6.77	6.50	9.25	7.30
multi_cnn_bilstm	6.09	5.77	6.14	5.17	7.70
resnet_bilstm	4.32	5.50	6.00	2.83	5.00
bilstm_resnet	3.64	4.32	2.41	1.50	1.10
FCN_bilstm	5.23	5.68	6.64	3.17	3.60
bilstm_FCN	4.14	4.09	2.36	2.50	2.50

Fuente: Elaboración propia, (2019).

De la Tabla 5.9, dada la cantidad de clases en los conjuntos de datos se destacan el rendimiento de los modelos bilstm_resnet y bilstm_FCN, diferenciándose significativamente del resto a medida que aumenta la cantidad de clases, mientras que de 2 a 5 clases se refleja un rendimiento similar entre todos los modelos.

Tabla 5.10: Promedio de ranking por cantidad de instancias del conjunto de datos.

	<201	201-1000	1001-3000	3001-5000	>5000
lstm_cnn	6.00	6.53	7.00	3.88	4.00
cnn_bilstm	5.06	6.47	6.45	5.81	6.50
cnn_lstm_separated	6.38	6.43	6.40	6.38	7.33
multi_cnn_bilstm2	7.06	7.30	6.65	8.31	4.83
cnn_lstm_separated2	6.13	7.40	7.70	6.25	7.33
multi_cnn_bilstm	6.31	6.20	5.55	6.13	6.50
resnet_bilstm	4.50	4.83	4.65	5.81	4.83
bilstm_resnet	4.69	2.03	2.35	3.13	4.00
FCN_bilstm	4.31	5.00	4.95	6.56	6.17
bilstm_FCN	4.56	2.80	3.30	2.75	3.50

Fuente: Elaboración propia, (2019).

A partir de la Tabla 5.10, se refleja la paridad que tienen todos los modelos cuando el número de instancias del conjunto de datos es menor o igual a 200, pero a medida que este aumenta se refleja un mejor rendimiento nuevamente de los modelos bilstm_resnet y bilstm_FCN, destacándose a su vez el rendimiento que llega a tener el modelo lstm_cnn que mejora notablemente su rendimiento comparado a los demás.

En cuanto al rendimiento de los modelos por la longitud de las series de tiempo, se destaca una vez más los modelos bilstm_resnet y bilstm_FCN con diferencias significativas por sobre los demás, a pesar de que tienen una baja de rendimiento en el rango de 451 – 700.

Tabla 5.11: Promedio de ranking por longitud de la serie de tiempo.

	<81	81-250	251-450	451-700	701-1000	>1000
lstm_cnn	4.375	6.58	6.13	5.38	5.25	5.20
cnn_bilstm	8.50	6.08	6.41	5.25	6.50	3.70
cnn_lstm_separated	6.63	6.35	6.94	5.63	6.75	5.70
multi_cnn_bilstm2	7.38	6.85	7.13	7.63	9.75	6.20
cnn_lstm_separated2	7.38	6.73	7.53	5.88	7.50	6.60
multi_cnn_bilstm	6.38	5.46	5.56	6.13	9.25	7.80
resnet_bilstm	3.63	4.42	5.16	4.38	3.25	7.50
bilstm_resnet	3.00	3.54	2.13	5.25	1.25	2.60
FCN_bilstm	4.13	4.77	5.59	4.38	3.75	7.40
bilstm_FCN	3.63	4.23	2.44	5.13	1.75	2.30

Fuente: Elaboración propia, (2019).

5.1.1 Test de hipótesis para la selección del mejor modelo

Una vez recopilados todos los resultados que presentan los modelos implementados, se procede a aplicar docimasia de hipótesis para seguir con la selección del mejor modelo. Por lo tanto, las hipótesis iniciales son:

- H_0 : No hay diferencias significativas entre los modelos para los conjuntos de datos considerados.
- H_1 : Existen diferencias significativas entre los modelos para los conjuntos de datos considerados.

A partir de ello se aplica el test de Friedman con la extensión de Iman y Davenport, sobre los resultados generales de los modelos con cada uno de los conjuntos de datos dada la métrica del coeficiente de Kappa registrados en la Tabla 5.1

De esta forma, se generan los siguientes resultados:

```
Iman Davenport's correction of Friedman's rank sum test
data: kappa
Corrected Friedman's chi-squared = 12.66, df1 = 9, df2 = 387, p-value < 2.2e-16
```

Figura 5.5: Test de hipótesis de Friedman con la extensión de Iman y Davenport para la métrica kappa de todos los modelos con todos los conjuntos de datos.

Fuente: Elaboración propia, (2019).

El p-valor que se muestra en la Figura 5.5 denota que hay al menos un algoritmo que funciona de manera diferente al resto y, por lo tanto, se acepta H_1 , por lo que es necesario proceder con un análisis por pares para comparaciones múltiples entre los modelos a través de un test post-hoc de los resultados.

Tests post-hoc

En primer lugar se aplica el test de Nemenyi

Nemenyi test

```
data: kappa
critical difference = 2.053, k = 10, df = 430
```

Figura 5.6: Test post-hoc de Nemenyi para la métrica kappa de todos los modelos con todos los conjuntos de datos.

Fuente: Elaboración propia, (2019).

Este procedimiento determina la diferencia crítica. Cualquiera de los dos algoritmos cuya diferencia de rendimiento es mayor que la diferencia crítica de 2,053 se considera significativamente diferente. Las múltiples comparaciones por pares resultantes del test se presentan a continuación, donde en negrilla se destacan los valores que hacen que existan diferencias significativas entre los modelos.

Tabla 5.12: Diferencias críticas entre cada uno de los modelos obtenidos por el test post-hoc de Nemenyi.

	lstm_cnn	cnn_bilstm	cnn_lstm_separated	multi_cnn_bilstm2	cnn_lstm_separated2	multi_cnn_bilstm	resnet_bilstm	bilstm_resnet	FCN_bilstm	bilstm_FCN
lstm_cnn	0.000	-0.205	-0.580	-1.239	-1.136	-0.193	0.977	2.966	0.659	2.614
cnn_bilstm	-0.205	0.000	-0.375	-1.034	-0.932	0.011	1.182	3.170	0.864	2.818
cnn_lstm_separated	-0.580	-0.375	0.000	-0.659	-0.557	0.386	1.557	3.545	1.239	3.193
multi_cnn_bilstm2	-1.239	-1.034	-0.659	0.000	0.102	1.045	2.216	4.205	1.898	3.852
cnn_lstm_separated2	-1.136	-0.932	-0.557	0.102	0.000	0.943	2.114	4.102	1.795	3.750
multi_cnn_bilstm	-0.193	0.011	0.386	1.045	0.943	0.000	1.170	3.159	0.852	2.807
resnet_bilstm	0.977	1.182	1.557	2.216	2.114	1.170	0.000	1.989	-0.318	1.636
bilstm_resnet	2.966	3.170	3.545	4.205	4.102	3.159	1.989	0.000	-2.307	-0.352
FCN_bilstm	0.659	0.864	1.239	1.898	1.795	0.852	-0.318	-2.307	0.000	1.955
bilstm_FCN	2.614	2.818	3.193	3.852	3.750	2.807	1.636	-0.352	1.955	0.000

Fuente: Elaboración propia, (2019).

La prueba de Nemenyi tiene la ventaja de tener un gráfico asociado para representar los resultados de la comparación, denominado gráfico de diferencia críticas, cuyo resultado se puede ver en la Figura 5.7.

En esta gráfica, cada algoritmo se coloca en un eje de acuerdo con su clasificación de ranking promedio. Luego, esos algoritmos que no muestran diferencias significativas se agrupan usando una línea horizontal. La gráfica también muestra el tamaño de la diferencia crítica requerida para considerar dos algoritmos como significativamente diferentes.

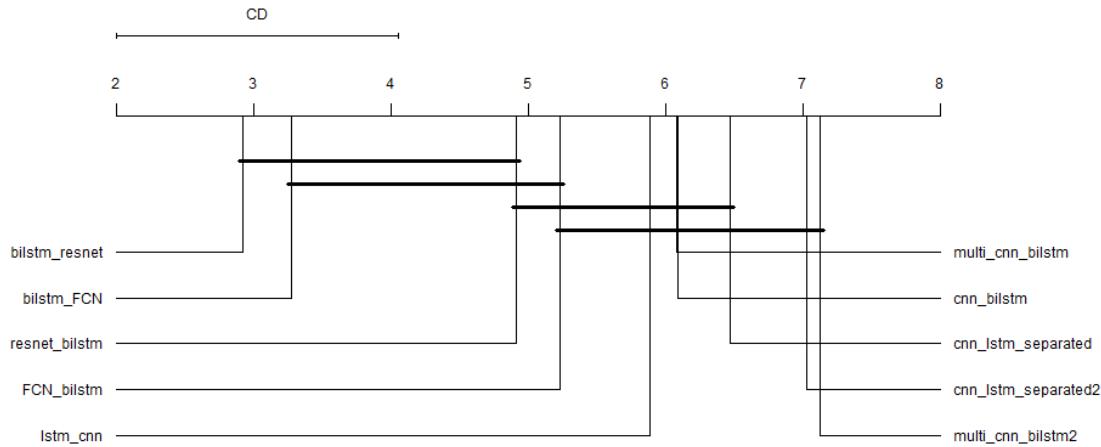


Figura 5.7: Gráfico de diferencias críticas basado en el test de Nemenyi entre todos los modelos.
Fuente: Elaboración propia, (2019).

De la Figura 5.7 y de la Tabla 5.12 se infiere con $\alpha = 0,05$ que no existen diferencias significativas entre:

- *bilstm_resnet* y *resnet_bilstm*
- *bilstm_FCN* y *FCN_bilstm*
- *resnet_bilstm* y *cnn_lstm_separated*
- *FCN_bilstm* y *multi_cnn_bilstm2*

Además se destaca a partir de la Figura 5.7 dados los valores de ranking promedio del test de Friedman, donde el valor más bajo representa a aquel modelo que presenta un mejor rendimiento, que en orden de mejor rendimiento a menor, los modelos se clasifican según la siguiente enumeración:

1. *bilstm_resnet*
2. *bilstm_FCN*
3. *resnet_bilstm*
4. *FCN_bilstm*
5. *lstm_cnn*
6. *cnn_bilstm*
7. *multi_cnn_bilstm*
8. *cnn_lstm_separated*

9. *multi_cnn_bilstm2*

10. *cnn_lstm_separated2*

Dichas observaciones se correlacionan altamente con las inferencias a priori establecidas en el apartado anterior, y los resultados presentados en las tablas 5.1 y 5.8.

A modo de comparación, se implementa el test post-hoc de Shaffer, que según la literatura es el que presenta mayor potencia con menor costo computacional, y siendo menos conservador que el test de Nemenyi, cuyos resultados se presentan a continuación. En negrita están subrayados aquellos modelos menores a 0,05, que indican que existen diferencias significativas entre los modelos.

Tabla 5.13: Corrección de Shaffer sobre los p-valores de las comparaciones entre los pares de modelos obtenidos por el test de Friedman

	lstm_cnn	cnn_bilstm	cnn_lstm_separated	multi_cnn_bilstm2	cnn_lstm_separated2	multi_cnn_bilstm	resnet_bilstm	bilstm_resnet	FCN_bilstm	bilstm_FCN
lstm_cnn	n/a	1.000	1.000	1.000	1.000	1.000	1.000	0.000	1.000	0.001
cnn_bilstm	1.000	n/a	1.000	1.000	1.000	1.000	1.000	0.000	1.000	0.000
cnn_lstm_separated	1.000	1.000	n/a	1.000	1.000	1.000	0.381	0.000	1.000	0.000
multi_cnn_bilstm2	1.000	1.000	1.000	n/a	1.000	1.000	0.017	0.000	0.092	0.000
cnn_lstm_separated2	1.000	1.000	1.000	1.000	n/a	1.000	0.031	0.000	0.130	0.000
multi_cnn_bilstm	1.000	1.000	1.000	1.000	1.000	n/a	1.000	0.000	1.000	0.000
resnet_bilstm	1.000	1.000	0.381	0.017	0.031	1.000	n/a	0.060	1.000	0.270
bilstm_resnet	0.000	0.000	0.000	0.000	0.000	0.000	0.060	n/a	0.010	1.000
FCN_bilstm	1.000	1.000	1.000	0.092	0.130	1.000	1.000	0.010	n/a	0.071
bilstm_FCN	0.001	0.000	0.000	0.000	0.000	0.000	0.270	1.000	0.071	n/a

Fuente: Elaboración propia, (2019).

A partir de los p-valores corregidos obtenidos del test, es posible obtener un gráfico alternativo al de diferencias críticas del test de Nemenyi, que representa como nodos conectados a los modelos que no muestran diferencias significativas entre si, a su vez que representa el mejor modelo en un nodo de color verde dados los rankings asignados a cada modelo en el test de Friedman.

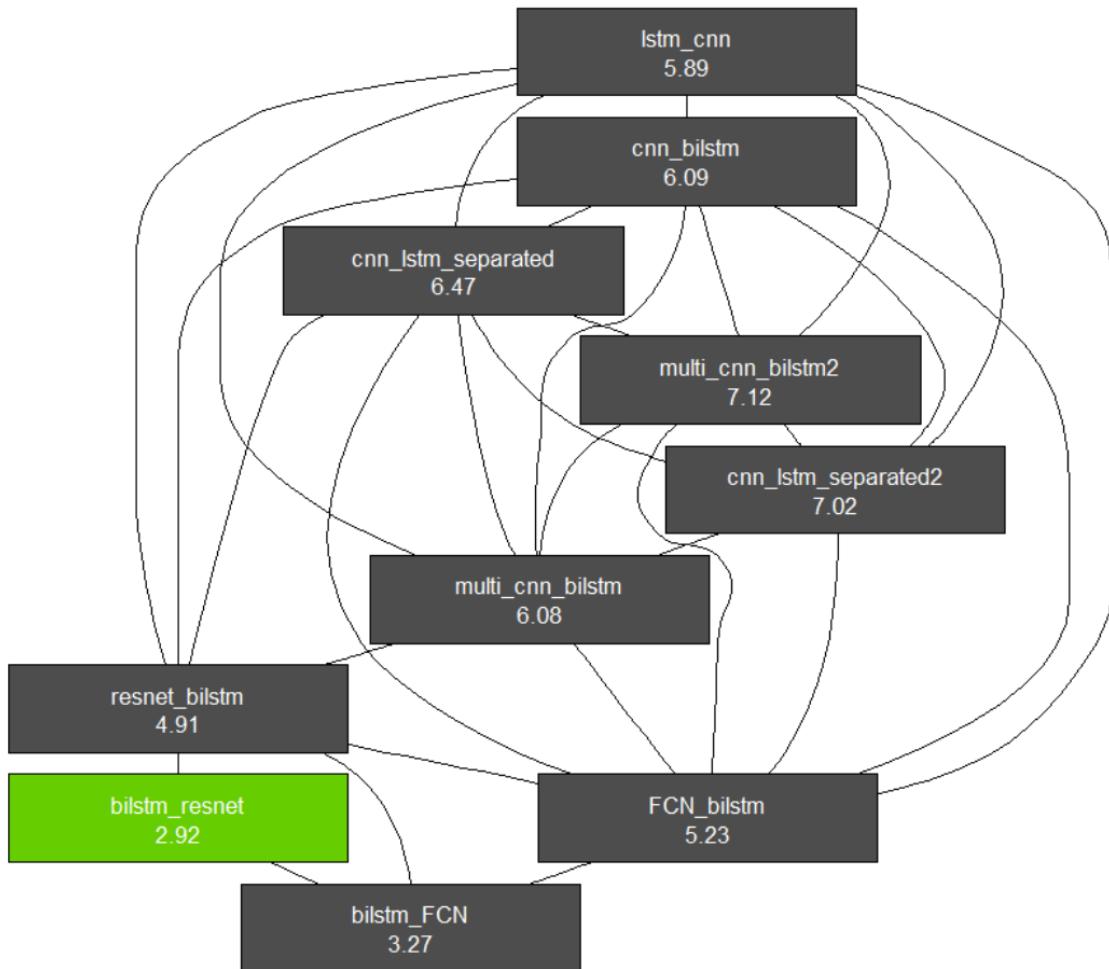


Figura 5.8: Gráfico de diferencias significativas entre los modelos basado en los p-valores obtenidos de la corrección de Bergman y Hommel con el test de Friedman.

Fuente: Elaboración propia, (2019).

De la Figura 5.8 y de la Tabla 5.13 es posible afirmar con $\alpha = 0,05$ que no existen diferencias significativas entre:

- *bilstm_resnet* y *resnet_bilstm*
- *bilstm_resnet* y *bilstm_FCN*
- *bilstm_FCN* y *resnet_bilstm*
- *bilstm_FCN* y *FCN_bilstm*
- *resnet_bilstm* y *lstm_cnn*
- *resnet_bilstm* y *multi_cnn_bilstm*
- *resnet_bilstm* y *cnn_bilstm*

- *resnet_bilstm* y *cnn_lstm_separated*
- *FCN_bilstm* y *lstm_cnn*
- *FCN_bilstm* y *multi_cnn_bilstm*
- *FCN_bilstm* y *cnn_bilstm*
- *FCN_bilstm* y *cnn_lstm_separated*
- *FCN_bilstm* y *cnn_lstm_separated2*
- *FCN_bilstm* y *multi_cnn_bilstm2*
- *lstm_cnn* y *multi_cnn_bilstm*
- *lstm_cnn* y *cnn_bilstm*
- *lstm_cnn* y *cnn_separated*
- *lstm_cnn* y *cnn_separated2*
- *lstm_cnn* y *multi_cnn_bilstm2*
- *cnn_bilstm* y *multi_cnn_bilstm*
- *cnn_bilstm* y *cnn_lstm_separated*
- *cnn_bilstm* y *cnn_lstm_separated2*
- *cnn_bilstm* y *multi_cnn_bilstm2*
- *multi_cnn_bilstm* y *cnn_lstm_separated*
- *multi_cnn_bilstm* y *cnn_lstm_separated2*
- *multi_cnn_bilstm* y *multi_cnn_bilstm2*
- *cnn_lstm_separated* y *cnn_lstm_separated2*
- *cnn_lstm_separated* y *multi_cnn_bilstm2*
- *cnn_lstm_separated2* y *multi_cnn_bilstm2*

En comparación con el test de Nemenyi, a través del test de Bergman y Hommel se observan mayor cantidad de diferencias no significativas entre los modelos, ofreciendo una visión más completa en la comparación de los resultados.

Se infiere a partir de ello que los modelos presentan resultados muy similares entre sí, ya que la mayoría de los modelos no presentan diferencias significativas con respecto a los

otros. Sin embargo se destacan los modelos que presentan mayor rendimiento, en específico los modelos bilstm_resnet y bilstm_FCN, quienes si presentan diferencias significativas con respecto a todos los modelos a excepción de aquellos con los que comparten una raíz similar, llaméntense los modelos resnet_bilstm y FCN_bilstm, quienes tienen un comportamiento no significativamente menor.

Cabe recordar que estos últimos 4 modelos mencionados, son implementaciones basadas en las arquitecturas FCN y ResNet encontradas en la literatura, y que ya de por sí presentan un buen rendimiento en los problemas de TSC.

Finalmente a partir de todos los resultados presentados anteriormente se seleccionan los modelos bilstm_resnet y bilstm_FCN para realizar las respectivas comparaciones con otros modelos de la literatura.

5.2 COMPARACIONES CON MODELOS CON ENFOQUE DE APRENDIZAJE PROFUNDO

A continuación se presentan los resultados generales para todos los modelos y los conjuntos de datos bajo la métrica del accuracy, obtenidos del estudio de Fawaz et al. (2019), que agrupa la revisión de los últimos modelos de la literatura enfocados en el aprendizaje profundo. Cabe recordar que el parámetro "Ganados" cuenta en cantidad las veces en que el modelo presenta el mejor rendimiento en cierto conjunto de datos, en comparación con los otros modelos a pesar de que haya un empate en rendimiento, y el "Ranking promedio" corresponde al promedio de ranking de Friedman.

Tabla 5.14: Accuracy promedio de diferentes modelos con enfoque de aprendizaje profundo de la literatura y de los mejores modelos implementados seleccionados.

	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCDCNN	Time-CNN	TWIESN	bilstm_FCN	bilstm_resnet
Adiac	0.397	0.844	0.829	0.484	0.022	0.020	0.610	0.379	0.416	0.860	0.834
Beef	0.720	0.697	0.753	0.643	0.200	0.200	0.563	0.763	0.537	0.833	0.7894
CBF	0.872	0.994	0.995	0.947	0.332	0.332	0.820	0.957	0.890	1.0	1.0
ChlorineConc	0.802	0.814	0.844	0.573	0.533	0.533	0.643	0.600	0.553	1.0	1.0
CinCECGtorso	0.840	0.824	0.826	0.911	0.381	0.250	0.736	0.745	0.300	0.997	0.998
Coffee	0.996	1.0	1.0	0.979	0.514	0.536	0.982	0.996	0.971	1.0	1.0
CricketX	0.591	0.792	0.791	0.694	0.189	0.074	0.495	0.552	0.622	0.842	0.848
CricketY	0.600	0.787	0.803	0.675	0.184	0.085	0.497	0.570	0.656	0.833	0.829
CricketZ	0.617	0.811	0.812	0.692	0.183	0.062	0.498	0.488	0.622	0.744	0.859
DiatomSizeRed	0.910	0.313	0.301	0.913	0.301	0.301	0.703	0.954	0.880	1.0	1.0
ECGFiveDays	0.970	0.987	0.975	0.982	0.499	0.497	0.762	0.882	0.698	1.0	1.0
FaceAll	0.793	0.945	0.839	0.793	0.170	0.080	0.717	0.768	0.657	0.990	0.997
FaceFour	0.840	0.928	0.955	0.815	0.268	0.295	0.712	0.906	0.855	1.0	1.0
FacesUCR	0.833	0.946	0.955	0.874	0.153	0.143	0.756	0.869	0.644	0.997	0.996
FiftyWords	0.684	0.627	0.740	0.723	0.220	0.125	0.589	0.621	0.496	0.816	0.805
Fish	0.848	0.958	0.979	0.866	0.134	0.126	0.758	0.849	0.875	0.962	0.967
GunPoint	0.927	1.0	0.991	0.936	0.513	0.493	0.867	0.932	0.961	1.0	1.0
Haptics	0.433	0.480	0.519	0.427	0.209	0.208	0.404	0.366	0.404	0.317	0.482
InlineSkate	0.337	0.339	0.373	0.292	0.167	0.165	0.215	0.287	0.330	0.564	0.482
ItalyPowerDemand	0.954	0.961	0.963	0.965	0.500	0.499	0.955	0.955	0.880	0.967	0.973
Lighting2	0.670	0.739	0.770	0.692	0.557	0.541	0.630	0.636	0.703	1.0	1.0
Lighting7	0.630	0.827	0.845	0.625	0.310	0.260	0.534	0.651	0.664	0.884	0.919
Mallat	0.918	0.967	0.972	0.876	0.135	0.123	0.901	0.920	0.596	1.0	1.0
MedicallImages	0.721	0.779	0.770	0.734	0.514	0.514	0.640	0.676	0.649	0.844	0.832
MoteStrain	0.858	0.937	0.928	0.840	0.508	0.539	0.765	0.882	0.785	0.976	0.984
NInvFetalECGThx1	0.916	0.956	0.945	0.916	0.161	0.029	0.905	0.865	0.494	0.961	0.971
NInvFetalECGThx2	0.917	0.953	0.946	0.932	0.160	0.029	0.915	0.898	0.525	0.948	0.953

OSULeaf	0.557	0.977	0.979	0.576	0.243	0.182	0.378	0.462	0.595	0.970	0.944
OliveOil	0.667	0.723	0.830	0.400	0.380	0.380	0.400	0.400	0.790	0.883	0.908
SonyAIBORobotSurf1	0.672	0.960	0.958	0.743	0.443	0.429	0.653	0.687	0.638	1.0	1.0
SonyAIBORobotSurf2	0.834	0.979	0.978	0.839	0.594	0.617	0.774	0.841	0.697	1.0	1.0
StarLightCurves	0.949	0.961	0.972	0.957	0.654	0.577	0.939	0.926	0.850	0.979	0.974
SwedishLeaf	0.851	0.969	0.956	0.930	0.118	0.065	0.846	0.884	0.825	0.953	0.953
Symbols	0.832	0.955	0.906	0.821	0.226	0.174	0.756	0.810	0.750	0.987	0.990
SyntheticControl	0.976	0.985	0.998	0.996	0.298	0.167	0.983	0.990	0.874	0.994	0.994
Trace	0.807	1.0	1.0	0.960	0.354	0.240	0.863	0.950	0.959	1.0	1.0
TwoLeadECG	0.762	1.0	1.0	0.863	0.500	0.500	0.760	0.872	0.852	1.0	1.0
TwoPatterns	0.946	0.871	1.0	1.0	0.403	0.259	0.978	0.992	0.871	1.0	1.0
UWaveGestureLibX	0.767	0.754	0.780	0.786	0.189	0.125	0.711	0.711	0.606	0.833	0.820
UWaveGestureLibY	0.698	0.639	0.670	0.696	0.237	0.121	0.636	0.626	0.520	0.789	0.743
UWaveGestureLibZ	0.697	0.726	0.750	0.711	0.180	0.121	0.650	0.642	0.565	0.798	0.742
Wafer	0.996	0.997	0.999	0.996	0.913	0.892	0.992	0.961	0.914	1.0	1.0
WordsSynonyms	0.598	0.564	0.622	0.613	0.284	0.219	0.463	0.566	0.490	0.824	0.822
Yoga	0.855	0.839	0.870	0.820	0.536	0.536	0.762	0.781	0.607	0.979	0.983
Ganados	0	6	8	1	0	0	0	0	0	27	27
Ranking promedio	6.227 (6)	3.988 (4)	3.318 (3)	5.432 (5)	10.148 (10)	10.807 (11)	7.852 (9)	6.693 (7)	7.705 (8)	1.989 (2)	1.841 (1)

Fuente: Fawaz et al. (2019), y elaboración propia, (2019).

A partir de la Tabla 5.14, es destacable el gran rendimiento que presentan los modelos bilstm_resnet y bilstm_FCN, con el mayor número de "ganados" y con la menor medida del ranking de Friedman, sobresaliendo más que los otros modelos. Ante dichos resultados, es importante considerar el rendimiento de cada uno de los modelos en cada uno de los dominios a los que pertenecen los conjuntos de datos, lo que se presenta en la Tabla 5.15.

Tabla 5.15: Accuracy promedio de los dominios de los conjuntos de datos para la comparación de modelos con el enfoque de aprendizaje profundo.

	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCD-CNN	Time-CNN	TWIESN	bilstm_FCN	bilstm_resnet
ECG	0.881	0.944	0.938	0.921	0.340	0.261	0.816	0.852	0.574	0.981	0.984
IMAGE	0.756	0.799	0.812	0.753	0.256	0.229	0.670	0.728	0.687	0.930	0.930
MOTION	0.630	0.703	0.721	0.657	0.228	0.162	0.553	0.575	0.587	0.747	0.756
SENSOR	0.821	0.952	0.954	0.851	0.539	0.508	0.789	0.843	0.787	0.977	0.982
SIMULATED	0.903	0.926	0.962	0.878	0.340	0.283	0.865	0.892	0.757	0.999	0.999
SPECTRO	0.758	0.891	0.911	0.733	0.319	0.306	0.641	0.740	0.701	0.934	0.911
TOTAL	0.792	0.869	0.883	0.799	0.337	0.292	0.722	0.772	0.682	0.928	0.927
Ranking	5.667 (5)	3.833 (4)	3.083 (3)	5.833 (6)	10.000 (10)	11.000 (11)	8.500 (9)	6.667 (7)	8.333 (8)	1.667 (2)	1.417 (1)

Fuente: Elaboración propia, (2019).

A partir de la Tabla 5.15, considerando los mejores rendimientos en cada dominio resaltados en negrita, se destaca que en todos los dominios sobresalen significativamente el modelo bilstm_FCN y principalmente el modelo bilstm_resnet. A su vez, estos modelos presentan el menor error promedio total de todos los conjuntos de datos, superando a sus modelos símiles originales ResNet y FCN.

5.2.1 Test de hipótesis para la comparación de los modelos de aprendizaje profundo

Para confirmar las diferencias entre los rendimientos de todos los modelos, se sigue el mismo procedimiento para la selección del mejor modelo implementado, por lo que se consideran las mismas hipótesis iniciales.

- H_0 : No hay diferencias significativas entre los modelos.
- H_1 : Existen diferencias significativas entre los modelos.

A partir de dichas hipótesis, se procede con la aplicación del test de Friedman con la extensión de Iman y Davenport, a través del cual se generan los siguientes resultados:

```
Iman Davenport's correction of Friedman's rank sum test

data: acc
Corrected Friedman's chi-squared = 221.65, df1 = 10, df2 = 430, p-value < 2.2e-16
```

Figura 5.9: Test de hipótesis de Friedman con la extensión de Iman y Davenport para la comparación con modelos con el enfoque de aprendizaje profundo.

Fuente: Elaboración propia, (2019).

El p-valor que se muestra en la Figura 5.9 refleja que hay al menos un modelo que funciona de manera diferente al resto, por lo que se acepta H_1 , y es necesario proceder con un test post-hoc.

Tests post-hoc

Como primer paso se aplica el test de Nemenyi de comparaciones múltiples por pares para determinar las diferencias críticas entre los modelos, y aprovechar la herramienta visual que presenta.

```
Nemenyi test

data: acc
Critical difference = 2.2872, k = 11, df = 473
```

Figura 5.10: Test post-hoc de Nemenyi para la comparación de los modelos con el enfoque de aprendizaje profundo.

Fuente: Elaboración propia, (2019).

En base al test, cualquier comparación por pares que presente un valor mayor a 2,2872, implica que existe una diferencia significativa entre ambos modelos.

La siguiente tabla presenta todas las comparaciones entre los modelos bajo el test de Nemenyi, destacando en negrita aquellos modelos en los que no existen diferencias significativas.

Tabla 5.16: Diferencias críticas del test de Nemenyi para las comparaciones de los modelos con el enfoque de aprendizaje profundo.

	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCD-CNN	Time-CNN	TWIESN	bilstm_FCN	bilstm_resnet
MLP	0.000	2.239	2.909	0.795	-3.920	-4.580	-1.625	-0.466	-1.477	4.239	4.386
FCN	2.239	0.000	0.670	-1.443	-6.159	-6.818	-3.864	-2.705	-3.716	2.000	2.148
ResNet	2.909	0.670	0.000	-2.114	-6.830	-7.489	-4.534	-3.375	-4.386	1.330	1.477
Encoder	0.795	-1.443	-2.114	0.000	-4.716	-5.375	-2.420	-1.261	-2.273	3.443	3.591
MCNN	-3.920	-6.159	-6.830	-4.716	0.000	-0.659	2.295	3.455	2.443	8.159	8.307
t-LeNet	-4.580	-6.818	-7.489	-5.375	-0.659	0.000	2.955	4.114	3.102	8.818	8.966
MCD-CNN	-1.625	-3.864	-4.534	-2.420	2.295	2.955	0.000	1.159	0.148	5.864	6.011
Time-CNN	-0.466	-2.705	-3.375	-1.261	3.455	4.114	1.159	0.000	-1.011	4.705	4.852
TWIESN	-1.477	-3.716	-4.386	-2.273	2.443	3.102	0.148	-1.011	0.000	5.716	5.864
bilstm_FCN	4.239	2.000	1.330	3.443	8.159	8.818	5.864	4.705	5.716	0.000	0.148
bilstm_resnet	4.386	2.148	1.477	3.591	8.307	8.966	6.011	4.852	5.864	0.148	0.000

Fuente: Wang et al. (2016), y elaboración propia, (2019).

A partir del gráfico de diferencias críticas presente en la Figura 5.11 y la Tabla 5.16, donde es posible detallar los resultados de la comparación por pares de todos los modelos con $\alpha = 0,05$, se puede establecer que los modelos bilstm_resnet y bilstm_FCN no presentan diferencias significativas entre sí, y a su vez no presentan diferencias significativas con los modelos ResNet y FCN, lo que refleja implícitamente las similitudes entre sus arquitecturas.

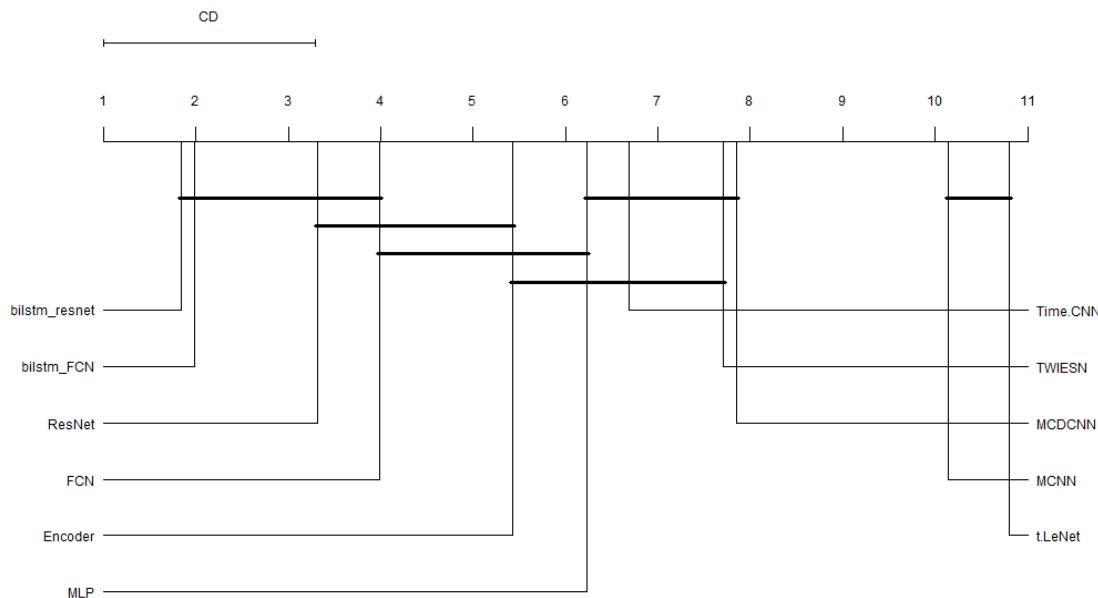


Figura 5.11: Gráfico de diferencias críticas basado en el test de Nemenyi para la comparación de los modelos con el enfoque de aprendizaje profundo.

Fuente: Elaboración propia, (2019).

Continuando con la comparación, se implementa el test post-hoc de Friedman con la corrección de Shaffer, cuyos resultados se presentan en la Tabla 5.17, donde se denotan las diferencias críticas entre los modelos en negrita, es posible afirmar que en concordancia con el test de Nemenyi, tampoco se esgrimen diferencias significativas entre los modelos bilstm_resnet y bilstm_FCN con los modelos ResNet y FCN, y a la vez se denotan diferencias significativas con el resto de modelos.

Tabla 5.17: Corrección de Shaffer sobre los p-valores del test de Friedman para la comparación de los modelos con el enfoque de aprendizaje profundo.

	MLP	FCN	ResNet	Encoder	MCNN	t-LeNet	MCD-CNN	Time-CNN	TWIESN	bilstm_FCN	bilstm_resnet
MLP	n/a	0.028	0.001	1.000	0.000	0.000	0.302	1.000	0.477	0.000	0.000
FCN	0.028	n/a	1.000	0.477	0.000	0.000	0.000	0.003	0.000	0.070	0.041
ResNet	0.001	1.000	n/a	0.045	0.000	0.000	0.000	0.000	0.000	0.601	0.477
Encoder	1.000	0.477	0.045	n/a	0.000	0.000	0.013	0.670	0.025	0.000	0.000
MCNN	0.000	0.000	0.000	0.000	n/a	1.000	0.023	0.000	0.012	0.000	0.000
t-LeNet	0.000	0.000	0.000	0.000	1.000	n/a	0.001	0.000	0.000	0.000	0.000
MCDCCNN	0.302	0.000	0.000	0.013	0.023	0.001	n/a	0.809	1.000	0.000	0.000
Time-CNN	1.000	0.003	0.000	0.670	0.000	0.000	0.809	n/a	1.000	0.000	0.000
TWIESN	0.477	0.000	0.000	0.025	0.012	0.000	1.000	1.000	n/a	0.000	0.000
bilstm_FCN	0.000	0.070	0.601	0.000	0.000	0.000	0.000	0.000	0.000	n/a	1.000
bilstm_resnet	0.000	0.041	0.477	0.000	0.000	0.000	0.000	0.000	0.000	1.000	n/a

Fuente: Elaboración propia, (2019).

Las inferencias anteriores se denotan con el gráfico de las diferencias no significativas a través de un grafo, que se presenta en la Figura 5.12.

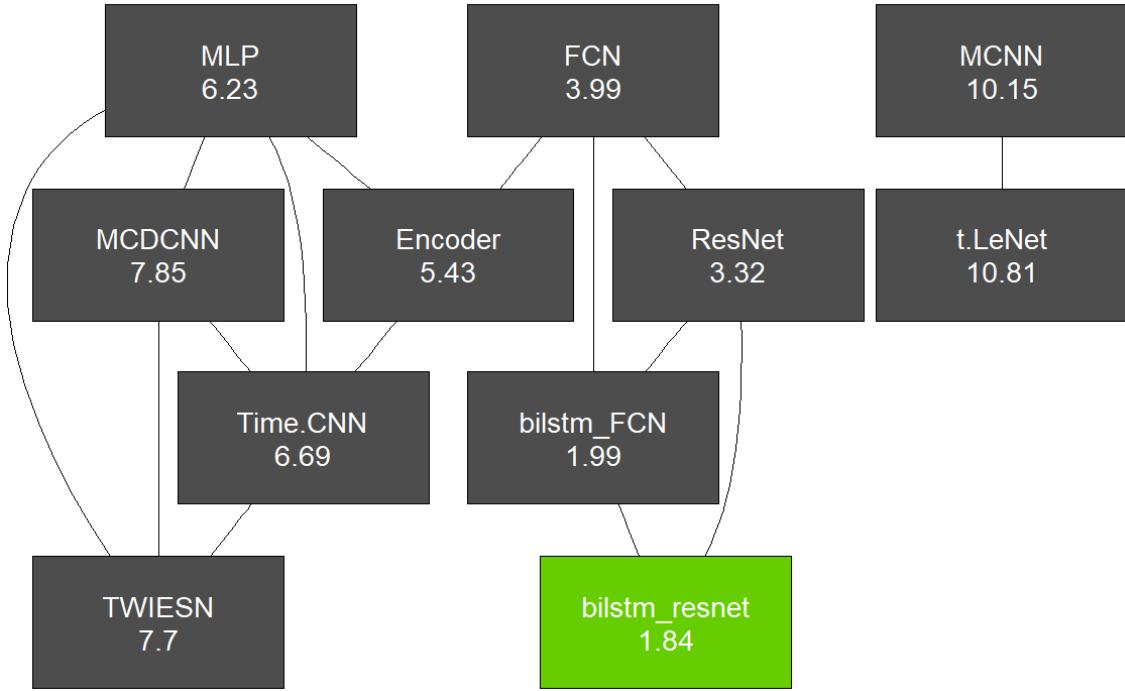


Figura 5.12: Gráfico de diferencias significativas entre los modelos con el enfoque de aprendizaje profundo basado en los p-valores obtenidos de la corrección de Shaffer con el test de Friedman.

Fuente: Elaboración propia, (2019).

En suma, del test de Nemenyi y el de Shaffer es posible afirmar con un $\alpha = 0,05$ que los modelos implementados en el presente trabajo no presentan diferencias significativas en rendimiento con los modelos ResNet y FCN. Cabe destacar que estos últimos presentan el mejor rendimiento entre los modelos con enfoque de aprendizaje profundo en tareas de TSC según la literatura.

Finalmente, a través de toda la información recopilada es posible establecer que en la comparativa de los modelos con enfoque de aprendizaje profundo, que los modelos bilstm_resnet y bilstm_FCN son los que presentan el mejor rendimiento.

En específico el ranking de los algoritmos en cuanto a rendimiento con la métrica del accuracy es el siguiente:

1. **bilstm_resnet**
2. **bilstm_FCN**
3. ResNet
4. FCN
5. Encoder
6. MLP

7. Time-CNN
8. TWIESN
9. MCDCNN
10. MCNN
11. t-LeNet

A partir de ello, se puede afirmar la supremacía en rendimiento bajo la métrica del accuracy que alcanzaron a nivel general los modelos híbridos de aprendizaje profundo implementados en el presente trabajo, lo que refleja el poder que presenta la combinación de arquitecturas convolucionales y recurrentes para afrontar los problemas de TSC, en comparación con enfoques puramente convolucionales o recurrentes.

5.3 COMPARACIONES CON OTROS MODELOS DE LA LITERATURA

A continuación se presentan los resultados generales bajo la métrica del accuracy para varios modelos con diversos enfoques, obtenidos del estudio de (Lines et al., 2018), en conjunto con los modelos aquí propuestos.

Cabe recordar que el parámetro "Ganados" cuenta en cantidad las veces en que el modelo presenta el mejor rendimiento en cierto conjunto de datos, y el "Ranking promedio" correspondiente al promedio de ranking de Friedman.

Tabla 5.18: Accuracy promedio de diferentes modelos de la literatura y de los mejores modelos implementados seleccionados.

	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	COTE	HIVE-COTE	bilstm_FCN	bilstm_resnet
Adiac	0.768	0.749	0.605	0.707	0.727	0.765	0.665	0.810	0.815	0.860	0.834
Beef	0.736	0.615	0.546	0.648	0.554	0.520	0.532	0.764	0.723	0.833	0.789
CBF	0.986	0.998	0.979	0.958	0.977	0.984	0.993	0.998	0.999	1.0	1.0
ChlorineConc	0.682	0.660	0.658	0.719	0.683	0.642	0.659	0.736	0.725	1.0	1.0
CinCECGtorso	0.918	0.900	0.714	0.974	0.716	0.743	0.946	0.983	0.988	0.998	0.998
Coffee	0.995	0.989	0.973	0.989	0.982	0.950	0.989	1.0	0.998	1.0	1.0
CricketX	0.777	0.764	0.769	0.691	0.731	0.696	0.801	0.814	0.830	0.842	0.848
CricketY	0.762	0.749	0.756	0.688	0.728	0.706	0.794	0.815	0.837	0.833	0.829
CricketZ	0.798	0.776	0.785	0.707	0.738	0.714	0.804	0.827	0.848	0.744	0.859
DiatomSizeRed	0.911	0.939	0.942	0.941	0.890	0.914	0.946	0.925	0.942	1.0	1.0
ECGFiveDays	0.955	0.983	0.907	0.922	0.849	0.840	0.847	0.986	0.990	1.0	1.0
FaceAll	0.968	0.974	0.963	0.949	0.942	0.962	0.976	0.990	0.996	0.990	0.997
FaceFour	0.794	0.996	0.909	0.891	0.862	0.889	0.879	0.850	0.950	1.0	1.0
FacesUCR	0.909	0.951	0.889	0.897	0.849	0.910	0.948	0.967	0.984	0.997	0.996
FiftyWords	0.713	0.702	0.748	0.728	0.744	0.776	0.821	0.801	0.807	0.816	0.805
Fish	0.974	0.969	0.931	0.807	0.913	0.912	0.913	0.962	0.976	0.962	0.967
GunPoint	0.999	0.994	0.964	0.962	0.964	0.972	0.974	0.992	0.997	1.0	1.0
Haptics	0.512	0.459	0.464	0.467	0.463	0.415	0.451	0.517	0.530	0.316	0.482
InlineSkate	0.393	0.503	0.382	0.378	0.377	0.449	0.476	0.526	0.526	0.564	0.513
ItalyPowerDemand	0.953	0.866	0.948	0.958	0.926	0.914	0.951	0.970	0.968	0.967	0.973
Lightning2	0.659	0.810	0.710	0.757	0.760	0.757	0.835	0.785	0.797	1.0	1.0
Lightning7	0.724	0.666	0.671	0.723	0.680	0.631	0.763	0.800	0.811	0.884	0.919
Mallat	0.972	0.949	0.929	0.937	0.951	0.908	0.961	0.974	0.975	1.0	1.0
MedicallImages	0.691	0.715	0.701	0.757	0.701	0.710	0.760	0.785	0.815	0.848	0.832
MoteStrain	0.882	0.846	0.891	0.874	0.886	0.917	0.875	0.902	0.947	0.976	0.984
NInvFetalECGThx1	0.947	0.841	0.877	0.880	0.842	0.807	0.849	0.929	0.932	0.961	0.971
NInvFetalECGThx2	0.954	0.904	0.898	0.914	0.862	0.826	0.914	0.946	0.952	0.948	0.953

OliveOil	0.881	0.870	0.864	0.883	0.864	0.892	0.879	0.901	0.898	0.883	0.908
OSULeaf	0.934	0.967	0.809	0.637	0.678	0.764	0.812	0.949	0.970	0.970	0.985
SonyAIBORobotSurf1	0.888	0.897	0.884	0.845	0.839	0.842	0.794	0.899	0.887	1.0	1.0
SonyAIBORobotSurf2	0.924	0.888	0.859	0.856	0.825	0.851	0.870	0.960	0.945	1.0	1.0
StarlightCurves	0.977	0.978	0.960	0.969	0.978	0.968	0.942	0.980	0.982	0.979	0.974
SwedishLeaf	0.938	0.918	0.886	0.892	0.908	0.926	0.916	0.967	0.969	0.953	0.953
Symbols	0.862	0.961	0.930	0.888	0.944	0.960	0.957	0.953	0.966	0.987	0.990
SyntheticControl	0.987	0.968	0.986	0.990	0.986	0.972	0.994	0.999	1.0	0.994	0.994
Trace	1.0	1.0	0.997	0.998	0.981	0.966	0.996	1.0	1.0	1.0	1.0
TwoLeadECG	0.984	0.984	0.958	0.842	0.910	0.928	0.958	0.982	0.993	1.0	1.0
TwoPatterns	0.952	0.991	1.0	0.990	0.974	0.967	1.0	1.0	1.0	1.0	1.0
UWaveGestureLibX	0.806	0.753	0.806	0.806	0.834	0.819	0.805	0.830	0.838	0.833	0.820
UWaveGestureLibY	0.737	0.661	0.717	0.727	0.746	0.753	0.730	0.766	0.776	0.789	0.743
UWaveGestureLibZ	0.747	0.695	0.736	0.741	0.776	0.767	0.726	0.760	0.778	0.798	0.742
Wafer	1.0	0.999	0.996	0.996	0.996	0.995	0.997	0.999	1.0	1.0	1.0
WordSynonyms	0.582	0.659	0.674	0.644	0.669	0.728	0.778	0.748	0.748	0.823	0.822
Yoga	0.822	0.910	0.863	0.867	0.834	0.874	0.885	0.898	0.917	0.979	0.983
Ganados	3	1	1	0	0	0	2	4	9	24	27
Ranking promedio	6.534 (5)	6.989 (7)	8.307 (9)	8.261 (8)	8.568 (11)	8.511 (10)	6.841 (6)	4.159 (4)	2.943 (3)	2.523 (2)	2.364 (1)

Fuente: Lines et al. (2018), y elaboración propia, (2019).

Comparando los modelos híbridos aquí propuestos con otros modelos de gran rendimiento bajo diversos enfoques de la literatura, es posible observar a partir de la Tabla 5.18, que los modelos bilstm_resnet y bilstm_FCN, presentan el mayor número de "ganados", sobresaliendo mucho más que los otros, además de su sobresaliente rendimiento en general expresado en su ranking promedio.

Cabe destacar que los modelos HIVE-COTE y su par COTE, son los que presentan mayor rendimiento general en tareas de TSC según la literatura, y a primera vista se ven sobrepasados en rendimiento por los modelos híbridos aquí propuestos.

Siguiendo las comparaciones, a continuación en la Tabla 5.19, se muestran los resultados generales para cada modelo, según el dominio al que pertenecen los distintos conjuntos de datos.

Tabla 5.19: Accuracy promedio de los dominios de los conjuntos de datos para la comparación con los modelos de la literatura.

	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	COTE	HIVE-COTE	bilstm_FCN	bilstm_resnet
ECG	0.952	0.922	0.871	0.906	0.836	0.829	0.903	0.965	0.971	0.981	0.984
IMAGE	0.836	0.878	0.835	0.816	0.820	0.853	0.866	0.893	0.912	0.937	0.936
MOTION	0.726	0.706	0.709	0.685	0.706	0.699	0.729	0.761	0.773	0.747	0.760
SENSOR	0.914	0.896	0.894	0.894	0.884	0.881	0.891	0.934	0.939	0.977	0.982
SIMULATED	0.916	0.913	0.910	0.919	0.914	0.895	0.921	0.941	0.940	0.999	0.999
SPECTRO	0.871	0.825	0.794	0.840	0.800	0.787	0.800	0.888	0.873	0.905	0.899
TOTAL	0.869	0.857	0.836	0.843	0.827	0.824	0.852	0.897	0.901	0.924	0.927
Ranking	6.000	6.917	8.750	8.083	9.167	10.167	6.917	3.333	3.000	1.917	1.750
	(5)	(6)	(9)	(8)	(10)	(11)	(6)	(4)	(3)	(2)	(1)

Fuente: Elaboración propia, (2019).

De la Tabla 5.19, se refleja ciertos problemas que presentan todos los modelos al afrontar principalmente el dominio MOTION, además de los dominios SPECTRO e IMAGE en menor medida. Teniendo en cuenta dicha consideración, se destaca que los modelos bilstm_resnet y bilstm_FCN son los que presentan un mayor promedio de accuracy en comparación con los otros, siendo el primero quien presenta un mejor rendimiento en general. Sin embargo, se debe destacar que en el dominio MOTION, ambos modelos son superados por HIVE-COTE, aunque no con una diferencia significativa.

En cuanto a la comparativa del rendimiento que presentan los modelos según las características de los conjuntos de datos, en las tabla 5.20, 5.21 y 5.22, se presenta los rankings promedios que presentan todos los modelos acorde al número de clases y de instancias, además de la longitud de la serie de tiempo, en donde un número de ranking más bajo indica un mejor rendimiento.

De la Tabla 5.20, es posible identificar que distintivamente los modelos bilstm_resnet

Tabla 5.20: Promedio de ranking por cantidad de clases del conjunto de datos para las comparaciones con los modelos de la literatura.

	2	3-5	6-10	11-20	21
ST-HESCA	6.09	6.95	7.00	6.17	6.0
BOSS	6.09	6.32	7.73	6.83	9.0
DTW_F	8.36	7.86	8.82	8.17	8.2
TSF	8.50	6.59	8.64	10.33	8.1
TSBF	9.14	8.64	7.32	9.67	8.6
LPS	8.95	9.00	7.73	8.67	8.0
EE	7.41	7.50	6.86	5.50	5.7
COTE	4.32	4.14	4.18	3.42	4.7
HIVE-COTE	3.68	3.23	2.18	2.00	3.5
bilstm_FCN	1.95	3.18	2.23	3.33	2.0
bilstm_resnet	1.50	2.59	3.32	1.92	2.2

Fuente: Elaboración propia, (2019).

y bilstm_FCN presentan el mejor rendimiento no importando el número clases, siendo siempre más que competitivo con respecto a los otros modelos.

Tabla 5.21: Promedio de ranking por cantidad de instancias del conjunto de datos para las comparaciones con los modelos de la literatura.

	<201	201-1000	1001-3000	3001-5000	>5000
ST-HESCA	6.31	6.47	7.25	6.75	4.50
BOSS	6.06	6.90	7.15	8.75	4.67
DTW_F	8.88	7.53	9.00	8.19	8.67
TSF	7.25	9.13	8.50	7.31	8.33
TSBF	8.88	9.13	9.15	6.75	7.83
LPS	8.94	8.80	7.80	8.00	9.67
EE	7.12	5.90	6.55	7.62	9.67
COTE	4.56	4.07	4.15	4.19	3.50
HIVE-COTE	3.94	2.60	2.70	2.81	3.17
bilstm_FCN	2.31	3.07	2.15	2.25	2.33
bilstm_resnet	1.75	2.40	1.60	3.38	3.67

Fuente: Elaboración propia, (2019).

En cuanto al rendimiento según la cantidad de instancias, Tabla 5.21, se denota un mayor rendimiento del modelo bilstm_resnet cuando el número de instancias es menor a 3000, mientras que bilstm_FCN se distingue cuando dicha cantidad es mayor. Así mismo se refleja la competitividad que ofrecen HIVE-COTE y COTE teniendo rankings no significativamente distintos.

Finalmente, en lo que se refiere a la longitud de la serie de tiempo, es posible observar que los mejores rendimientos los poseen una vez más los modelos bilstm_FCN y bilstm_ResNet, a excepción de cuando la longitud de la serie de tiempo es mayor a 1000, en donde sobresalen distintivamente COTE y HIVE-COTE.

Tabla 5.22: Promedio de ranking por longitud de la serie de tiempo para las comparaciones con los modelos de la literatura.

	<81	81-250	251-450	451-700	701-1000	>1000
ST-HESCA	5.75	6.54	7.66	6.00	2.00	5.8
BOSS	8.00	6.35	7.41	5.75	9.00	6.7
DTW_F	7.88	8.62	7.72	9.12	8.00	9.2
TSF	7.00	8.92	8.69	7.75	6.25	7.4
TSBF	9.62	9.08	8.00	8.50	9.50	7.9
LPS	9.75	8.31	7.88	8.38	11.00	9.2
EE	7.25	7.15	6.06	7.38	7.25	7.6
COTE	2.50	4.23	4.81	4.12	5.00	2.9
HIVE-COTE	3.50	2.92	2.88	3.50	3.50	2.1
bilstm_FCN	2.75	1.96	2.31	3.38	3.00	3.6
bilstm_resnet	2.00	1.92	2.59	2.12	1.50	3.6

Fuente: Elaboración propia, (2019).

5.3.1 Test de hipótesis para la comparación con modelos de la literatura

Para confirmar las diferencias entre los rendimientos de todos los modelos, se sigue el mismo procedimiento que etapas anteriores, donde las hipótesis son:

- H_0 : No hay diferencias significativas entre los modelos.
- H_1 : Existen diferencias significativas entre los modelos.

Luego con la aplicación del test de Friedman con la extensión de Iman y Davenport, cuyos resultados se reflejan en la Figura 5.13, es posible establecer que dado el p-valor, hay al menos un modelo que presenta diferencias significativas con respecto al resto, por lo que se acepta H_1 , y es necesario proceder con un test post-hoc.

```
Iman Davenport's correction of Friedman's rank sum test

data: acc
Corrected Friedman's chi-squared = 58.404, df1 = 10, df2 = 430, p-value < 2.2e-16
```

Figura 5.13: Test de hipótesis de Friedman con la extensión de Iman y Davenport para la comparación con modelos de la literatura.

Fuente: Elaboración propia, (2019).

Tests post-hoc

Siguiendo la tónica, se aplica el test de Nemenyi de comparaciones múltiples por pares para determinar las diferencias críticas entre los modelos, de lo que se puede obtener los resultados de la Tabla 5.14.

Nemenyi test

```
data: acc
Critical difference = 2.2872, k = 11, df = 473
```

Figura 5.14: Test post-hoc de Nemenyi para la comparación con modelos de la literatura
Fuente: Elaboración propia, (2019).

En base al test, cualquier comparación por pares que presente un valor mayor a 2,2872, involucra la presencia de una diferencia significativa entre ambos modelos, a partir de lo cual, es posible distinguir con negrita en la Tabla 5.23, todos los modelos en los que se reflejan diferencias significativas en rendimiento.

Tabla 5.23: Diferencias críticas del test de Nemenyi para las comparaciones con los modelos de la literatura.

	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	COTE	HIVE-COTE	bilstm_FCN	bilstm_resnet
ST-HESCA	0.000	-0.455	-1.773	-1.727	-2.034	-1.977	-0.307	2.375	3.591	4.011	4.170
BOSS	-0.455	0.000	-1.318	-1.273	-1.580	-1.523	0.148	2.830	4.045	4.466	4.625
DTW_F	-1.773	-1.318	0.000	0.045	-0.261	-0.205	1.466	4.148	5.364	5.784	5.943
TSF	-1.727	-1.273	0.045	0.000	-0.307	-0.250	1.420	4.102	5.318	5.739	5.898
TSBF	-2.034	-1.580	-0.261	-0.307	0.000	0.057	1.727	4.409	5.625	6.045	6.205
LPS	-1.977	-1.523	-0.205	-0.250	0.057	0.000	1.670	4.352	5.568	5.989	6.148
EE	-0.307	0.148	1.466	1.420	1.727	1.670	0.000	2.682	3.898	4.318	4.477
COTE	2.375	2.830	4.148	4.102	4.409	4.352	2.682	0.000	1.216	1.636	1.795
HIVE-COTE	3.591	4.045	5.364	5.318	5.625	5.568	3.898	1.216	0.000	0.420	0.580
bilstm_FCN	4.011	4.466	5.784	5.739	6.045	5.989	4.318	1.636	0.420	0.000	0.159
bilstm_resnet	4.170	4.625	5.943	5.898	6.205	6.148	4.477	1.795	0.580	0.159	0.000

Fuente: Wang et al. (2016), y elaboración propia, (2019).

El gráfico de diferencias críticas presente en la Figura 5.15, refleja en mayor medida las distinciones entre los modelos, con lo que es posible detallar que los modelos bilstm_resnet y bilstm_FCN de mejor rendimiento, no presentan diferencias significativas con COTE y HIVE-COTE, confirmando cierta paridad que se venía arrastrando a partir de los resultados anteriores. Estos cuatro modelos en general son distintivamente mejores en rendimiento a los modelos más próximos ST-HESCA y EE, y muy significativamente diferentes a los otros modelos.

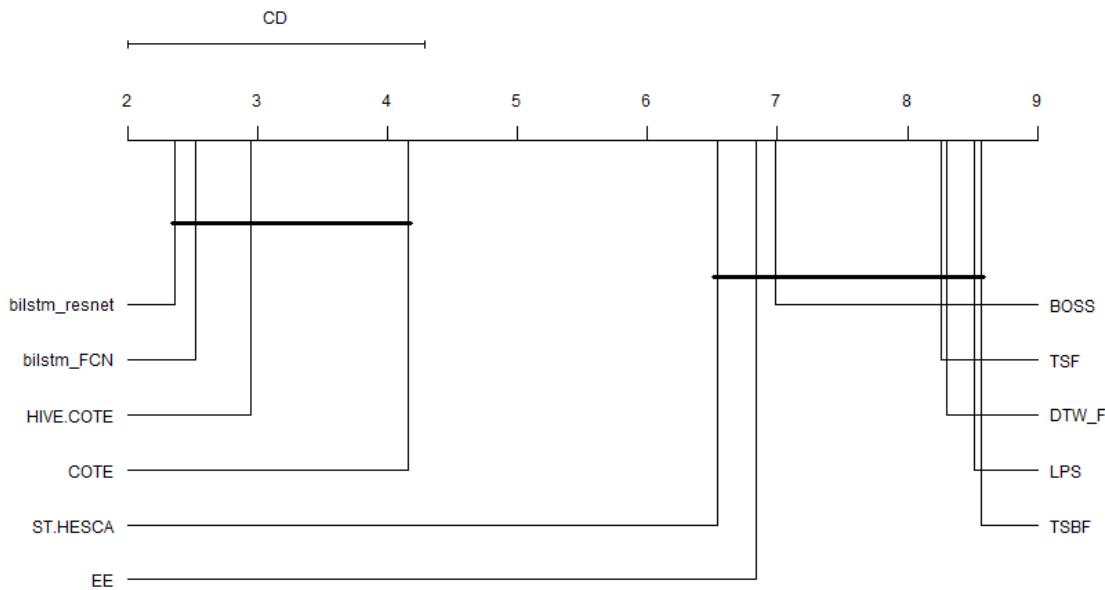


Figura 5.15: Gráfico de diferencias críticas basado en el test de Nemenyi para la comparación con modelos de la literatura.

Fuente: Elaboración propia, (2019).

Continuando con la comparación, se implementa el test post-hoc de Friedman con la corrección de Shaffer, cuyos resultados se presentan en la Tabla 5.24, donde se denotan las diferencias críticas entre los modelos en negrita, es posible afirmar que en concordancia con el test de Nemenyi, tampoco se esgrimen diferencias significativas entre los modelos bilstm_resnet y bilstm_FCN, con COTE y HIVE-COTE, a pesar de que si se denotan ciertas diferencias con respecto a COTE,

Tabla 5.24: Corrección de Shaffer sobre los p-valores del test de Friedman para la comparación con modelos de la literatura.

	ST-HESCA	BOSS	DTW_F	TSF	TSBF	LPS	EE	COTE	HIVE-COTE	bilstm_FCN	bilstm_resnet
ST-HESCA	n/a	1.000	0.292	0.335	0.109	0.129	1.000	0.022	0.000	0.000	0.000
BOSS	1.000	n/a	0.934	1.000	0.484	0.563	1.000	0.002	0.000	0.000	0.000
DTW_F	0.292	0.934	n/a	1.000	1.000	1.000	0.649	0.000	0.000	0.000	0.000
TSF	0.335	1.000	1.000	n/a	1.000	1.000	0.713	0.000	0.000	0.000	0.000
TSBF	0.109	0.484	1.000	1.000	n/a	1.000	0.335	0.000	0.000	0.000	0.000
LPS	0.129	0.563	1.000	1.000	1.000	n/a	0.381	0.000	0.000	0.000	0.000
EE	1.000	1.000	0.649	0.713	0.335	0.381	n/a	0.004	0.000	0.000	0.000
COTE	0.022	0.002	0.000	0.000	0.000	0.000	0.004	n/a	1.000	0.413	0.278
HIVE-COTE	0.000	1.000	n/a	1.000	1.000						
bilstm_FCN	0.000	0.413	1.000	n/a	1.000						
bilstm_resnet	0.000	0.278	1.000	1.000	n/a						

Fuente: Elaboración propia, (2019).

Todas estas observaciones preliminares se continúan reflejando en el grafo de diferencias significativas de la Figura 5.16.

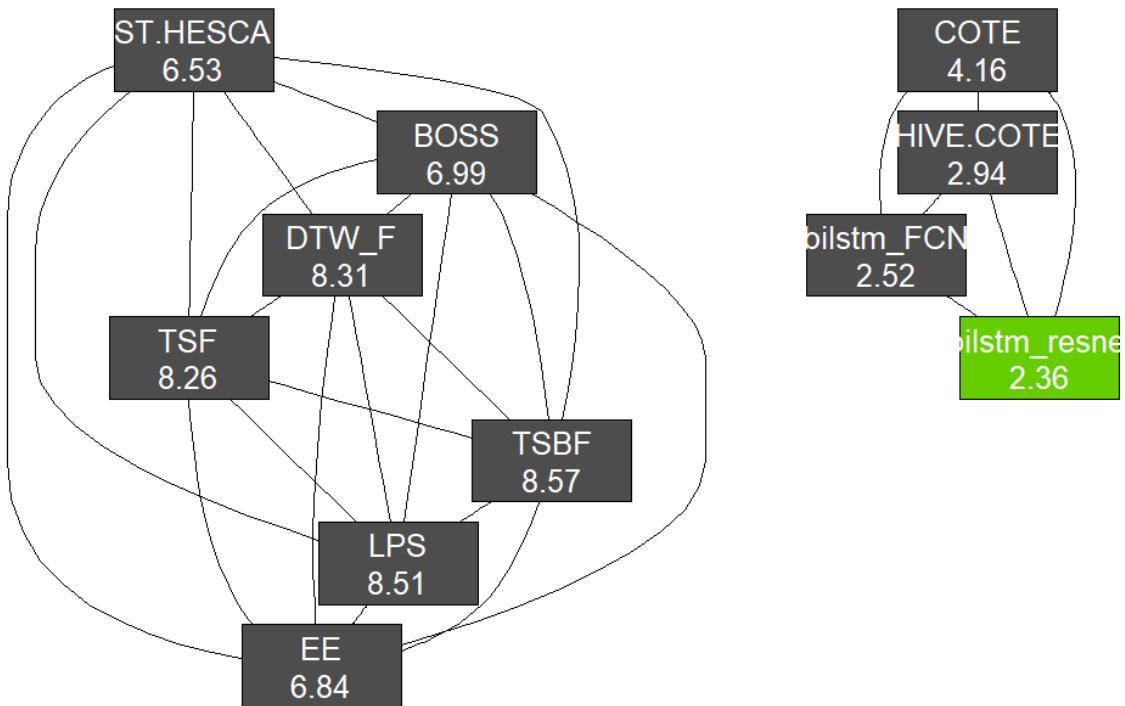


Figura 5.16: Gráfico de diferencias significativas para las comparaciones con los modelos de la literatura. basado en los p-valores obtenidos de la corrección de Shaffer con el test de Friedman.

Fuente: Elaboración propia, (2019).

Finalmente del test de Nemenyi y el de Shaffer es posible afirmar con un $\alpha = 0,05$ que los modelos implementados en el presente trabajo no presentan diferencias significativas con los modelos de mayor rendimiento de la literatura, presentando incluso resultados de vanguardia en lo referente a tareas de TSC.

El orden de los modelos en cuanto al rendimiento final corresponde a:

1. **bilstm_resnet**
2. **bilstm_FCN**
3. HIVE-COTE
4. COTE
5. ST-HESCA
6. EE
7. BOSS
8. TSF
9. DTW_F

10. LPS

11. TSBF

Para concluir esta etapa, se puede decir que los modelos híbridos de aprendizaje profundo implementados en el presente trabajo, alcanzaron un rendimiento de vanguardia en los problemas de TSC, superando a modelos mucho más complicados como HIVE-COTE y COTE, lo que refleja el poder que puede llegar a presentar las arquitecturas híbridas de aprendizaje profundo, específicamente la combinación entre CNN Y LSTM, en la clasificación de series de tiempo.

CAPÍTULO 6. DISCUSIÓN

6.1 SELECCIÓN DEL MEJOR MODELO IMPLEMENTADO

La primera inferencia teórica tras la aplicabilidad del enfoque híbrido CNN-LSTM, sugería que mientras la red convolucional realizaba la extracción de características, la red recurrente aprendía las dependencias temporales de las series de tiempo. Una vez realizada una evaluación experimental de varios modelos bajo el enfoque mencionado, se tienen ahora las bases para realizar los análisis correspondientes que permitan obtener las conjeturas pertinentes al caso.

Para la construcción inicial de las arquitecturas se realizaron montajes empíricos buscando obtener la mejor configuración que obtuviese el mejor rendimiento. Los montajes incluyeron dos tipos de enfoques estructurales, lineal y por raíces. Para todos los montajes se alternó el orden de las capas LSTM y CNN, buscando dirimir si era mejor extraer en primer lugar las dependencias temporales a través de las capas recurrentes, o si en su defecto era mejor realizar la extracción de características con las capas convolucionales como primer proceso. Dicha búsqueda se evidencia mayormente en el enfoque lineal, tal como se observa en los modelos cnn_bilstm (Fig. 4.1), lstm_cnn (Fig. 4.2), FCN_bilstm (Fig. 4.7), bilstm_FCN (Fig. 4.8), resnet_bilstm (Fig. 4.9) y bilstm_resnet (Fig. 4.10) mientras que en el enfoque por raíces se buscaba explotar el rendimiento de cada capa por separado como los modelo cnn_lstm_separated (Fig. 4.3) y cnn_lstm_separated2 (Fig. 4.6), la explotación de capas convolucionales separadas para luego conjuntamente pasar a la capa recurrente como es el caso del modelo multi_cnn_bilstm2 (Fig. 4.4), o la concatenación de varios enfoques lineales como lo hace el modelo multi_cnn_bilstm (Fig. 4.1). Finalmente los resultados reflejan que el enfoque lineal presenta en general un mayor rendimiento sobre el enfoque por raíces, considerando especialmente el rendimiento que presentan los modelos según las características de los conjuntos de datos, en donde se puede dilucidar que la complejidad del modelo en términos de su estructura, no garantiza los mejores resultados cuando se complejizan las características de los conjuntos.

Específicamente dentro de los modelos lineales se evidencia un mayor rendimiento en los modelos en que implementan una capa recurrente LSTM como primer proceso por sobre los que implementan capas convolucionales al inicio de la arquitectura, demostrado en los modelos bilstm_resnet y bilstm_FCN, que son los que finalmente alcanzan el mayor rendimiento. Por lo tanto, es posible afirmar que la extracción de las dependencias a largo plazo con una capa LSTM, originalmente diseñada para trabajar con series de tiempo, para luego continuar con la extracción de características a través de las capas convolucionales, que de por sí solas han presentado un gran rendimiento en múltiples áreas, presenta mejor rendimiento que un diseño arquitectural en el otro sentido. Sin embargo, se debe aclarar que los distintos tests de hipótesis reflejaron que

no existían diferencias significativas entre todos los modelos, especialmente entre los homólogos bilstm_resnet y resnet_bilstm, y bilstm_FCN y FCN_bilstm, enfoques lineales que se diferencian entre sí solo por el orden de las capas, por lo que es necesario seguir realizando más pruebas con más conjuntos de datos para dirimir mayores diferencias.

Es interesante rescatar el rendimiento que posee el modelo lstm_cnn, que solo consta de una capa bilstm y una capa convolucional, a diferencia de la complejidad que alcanzan a llegar otros modelos con mayores números de capas y cantidad de filtros, y estructuras más complejas. En total alcanza a tener mejor rendimiento en 15 conjuntos de datos y no posee diferencias significativas bajo la métrica kappa con los modelos resnet_bilstm y FCN_bilstm, quienes lo superan un poco en rendimiento. Su simplicidad contribuye a que el tiempo de entrenamiento sea el menor entre todos los modelos, por lo que sería interesante modificar los hiperparámetros del modelo para tratar de conseguir mejores resultados. Sin embargo, cabe señalar que en etapas tempranas del presente estudio, al comparar su desempeño con los modelos de la literatura, no alcanzaba a ser lo suficientemente competitivo, lo que finalmente llevó a continuar la búsqueda de otros modelos que pudiesen presentar mejor desempeño.

Ahondando en las características de los conjuntos de datos, llama la atención el bajo rendimiento que obtuvieron todos los modelos en los conjuntos de datos Haptics e InlineSkate. Incluso en los modelos de la literatura se refleja un menor rendimiento en comparación a los otros conjuntos de datos. Si bien se podría decir que ambos conjuntos son complejos en sus características, presentando 5 y 7 clases respectivamente, con una longitud de la series de tiempo mayor a 1000 registros y presentando no más de 650 instancias para distribuirlas en muestras para entrenamiento y prueba, muchos conjuntos de datos presentan igual o mayor complejidad y sin embargo, es posible obtener un buen rendimiento sobre ellos con cualquiera de los modelos. Por lo tanto, una explicación lógica se basa en el dominio al cual pertenece ambos conjuntos de datos, ya que en el dominio MOTION es aquel en el cual todos los modelos presentan el menor rendimiento.

En específico el conjunto de datos Haptics contempla los datos del eje X de electrodos que registran los movimientos de la mano durante el ingreso de una contraseña en un panel táctil, mientras que InlineSkate recopila datos de la medida angular del tobillo de atletas en pruebas de patinaje de velocidad. Ambos presentan un alto nivel de complejidad en su estudio de origen, y naturalmente están susceptibles al ruido y a las condiciones en que se realizan los movimientos. Otros conjuntos de datos donde también se evidencian problemas de rendimiento son CricketX, CricketY, CricketZ, UWaveGestureLibraryX, UWaveGestureLibraryY y UWaveGestureLibraryZ, también pertenecientes al mismo dominio, y por lo tanto, con similares condiciones.

En la misma linea, en los dominios IMAGE y SPECTRO se evidencian problemas de rendimiento,

sin embargo, estos son menores a los que se registran en MOTION. Bajo dichos dominios se destacan los modelos bilstm_resnet y bilstm_FCN con los mejores rendimientos y menores tasas de error, aunque específicamente en el dominio SPECTRO ambos modelos presentan menor rendimiento en comparación a la mayoría de los otros modelos.

Los resultados en el rendimiento en este dominio, en particular con los conjuntos de datos Beef y OliveOil, se pueden explicar por el bajo número de instancias, 60 en total, con 5 y 4 clases respectivamente y una longitud de las series de tiempo de 470 y 570 respectivamente, lo que deja un pequeño margen para la generalización y la extracción de características. Por lo tanto, es posible concluir que a medida que aumentan los filtros y la complejidad en la arquitectura, se necesitan más muestras para alcanzar la generalización, lo que explica el bajo rendimiento que alcanzan modelos más complejos como aquellos que implementan más raíces, o modelos complejos como los que implementan arquitecturas resnet y FCN, por sobre aquellos enfoques lineales más simples como cnn_bilstm y lstm_cnn quienes alcanzan el mayor rendimiento en este dominio.

Bajo los dominios ECG, SENSOR y SIMULATED, todos los modelos en general presentan un buen rendimiento, diferenciándose en cierta medida en el nivel de error al que alcanzan a llegar, en donde los modelos bilstm_resnet y bilstm_FCN son los que se destacan. Cabe señalar que los conjuntos de datos dentro de estos dominios presentan el suficiente nivel de muestras y largo de las series de tiempo como para poder alcanzar una buena generalización que permita obtener los resultados vistos. Sólo se escapan algunos conjuntos de datos como Lightning7 del dominio SENSOR, que solo presenta 143 instancias, un largo de 319 para la serie de tiempo y 7 clases, lo que deja poco margen para la generalización, que se evidencia en el bajo rendimiento que obtienen la mayoría de los modelos en este conjunto de datos.

Por último, es posible establecer en cuanto a las características de los conjuntos de datos, que ni la longitud de las series de tiempo, ni la cantidad de clases por sí solas proporcionan mayor información sobre el rendimiento de los modelos, mientras que el número de instancias prima para lograr altas precisiones, afirmación que se evidencia en la literatura. Además es posible establecer que la combinación de todos los factores inciden en el rendimiento, puesto que si existen pocas muestras, pero la cantidad de clases es menor a 5 y la longitud de las series de tiempo es baja, se puede alcanzar una buena generalización; mientras que un alto número de clases y una longitud alta de la serie de tiempo puede incidir en un mal rendimiento de los modelos, cuando se tienen pocas instancias para el entrenamiento.

Para concluir el proceso de la selección del mejor modelo, a pesar de que la idea inicial era contemplar uno solo, finalmente se optó por elegir dos de ellos. La decisión se basó en primer lugar en que prácticamente no existe ninguna diferencia entre ambos modelos tanto en la métrica del coeficiente de kappa, como bajo la métricas de error y accuracy, cuyos resultados

se presentan en los apéndices A y B. En segundo lugar, considerando los rankings promedios para todos los conjuntos de datos, se evidencia que bajo las métricas kappa y accuracy, el modelo bilstm_resnet se lleva el primer lugar seguido de bilstm_FCN, sin embargo, considerando la métrica del error, el orden se da en el otro sentido. Y en tercer lugar, considerando la cantidad de "ganados", veces en que el modelo obtiene el mejor rendimiento, a pesar de que bajo las métricas kappa y accuracy, se obtiene una diferencia de aproximadamente diez veces a favor del modelo bilstm_resnet, bajo la métrica del error, el resultado es muy parejo.

6.2 COMPARACIONES CON MODELOS CON ENFOQUE DE APRENDIZAJE PROFUNDO

Durante el último tiempo se han multiplicado los estudios que proponen modelos con enfoque de aprendizaje profundo para afrontar problemas de TSC, presentando un rendimiento destacable en general. A pesar de ello, estos aún no han sido explotados en su totalidad con todas las posibilidades que ofrecen, lo que se refleja en su no inclusión en el estudio de Bagnall et al. (2017), que suponía hacer una evaluación experimental de los últimos mejores modelos propuestos. Particularmente para la comparación se incluye el modelo MCNN propuesto en Cui et al. (2016), y los modelos FCN, ResNet y MLP propuestos en Wang et al. (2016), todos con un buen rendimiento en tareas de TSC, además de los modelos Encoder, Time-CNN, TWIESN, MCDCNN, y t-LeNet.

En cuanto a rendimiento los que mas sobresalen son FCN y ResNet, a partir de los cuales se realizan implementaciones híbridas agregando capas LSTM buscando comprobar la hipótesis en la que se sustenta el presente estudio, y que finalmente son las que presentan el mejor rendimiento tras la etapa anterior.

Era esperable encontrar que los modelos ResNet y FCN alcanzaran un rendimiento similar al de sus híbridos bilstm_resnet y bilstm_FCN dado el ranking promedio y el promedio de accuracy obtenido en todos los dominios, lo que claramente se debe a la semejanza de sus arquitecturas. Sin embargo, a pesar de que no son significativas, existen pequeñas diferencias que hacen sobresalir a los modelos híbridos por sobre sus símiles. Esto se puede deber a la forma en que realizan la detección de las zonas discriminativas de la serie de tiempo que inciden en la decisión de la clase, lo que se interrelaciona directamente con las características de cada conjunto de datos. Bajo dicha afirmación se puede suponer que en las series de tiempo, la capa LSTM puede aprender qué observaciones han visto anteriormente y cómo son relevantes para encontrar la zona discriminativa, información que va propagando a las capas convolucionales para la extracción de características, las que a su vez extraen aquellas que son invariantes en el tiempo.

Para ahondar en esta comparación, se utiliza la visualización de los mapas de activación sobre

la capa Global Average Pooling (GAP), lo que permite identificar qué regiones de una serie de tiempo de entrada constituyen la razón de una determinada clasificación.

Específicamente se visualiza los mapas de activación resultantes del entrenamiento del conjunto de datos de GunPoint (Véase las figuras 6.1 y 6.2), introducido por Ratanamahatana & Keogh (2005) como un problema de TSC. Este conjunto pertenece al dominio MOTION, e involucra a un actor masculino y uno femenino que realizan dos acciones Gun-Draw y Point. Para Gun-Draw (Clase 1), el movimiento es el siguiente: los actores tienen primero las manos por los lados, luego sacan un arma de una funda puesta en sus cadera, la apuntan hacia el objetivo durante un segundo, y finalmente la colocan en su funda y retornan sus manos a la posición inicial. Al igual que en Gun-Draw, para Point (Clase 2), los actores siguen los mismos pasos, pero en lugar de apuntar con un arma, apuntan con su dedo índice. Para el conjunto de datos se contiene solo una serie de tiempo del eje x de los movimientos, puesto que el rastreo del centroide de las manos derechas de los actores en los ejes X e Y eran altamente correlacionados (Fawaz et al., 2019). Se utiliza GunPoint para la visualización debido a que solo presenta dos clases, y es equitativo en términos del rendimiento que obtienen los modelos, con un error cercano a 0 y un accuracy de casi 1.

A partir del estudio de Fawaz et al. (2019), en el que se exploran diversos modelos de aprendizaje profundo en tareas de TSC, se obtienen los mapas de activación de los modelos ResNet y FCN. Basado en sus inferencias, de las figuras 6.1 y 6.2, el análisis dicta que todos los modelos están descuidando las regiones no discriminatorias de la meseta al tomar la decisión de clasificación, representado por las partes planas azules de la serie de tiempo que indica que no hay contribución a la decisión del clasificador. En cuanto a las regiones altamente discriminativas (las regiones roja y amarilla), se detallan diferencias entre todos los clasificadores. Para la clase 1, bilstm_FCN detecta una zona discriminativa más amplia en todo el momento en que el actor levanta su brazo con el arma, mientras que los otros modelos solo consideran el instante de inicio de dicha acción. Así mismo, los modelos híbridos detectan una zona discriminativa más corta que ResNet y FCN, para el momento en que finaliza la acción de dejar el arma en la funda. Para la clase 2, se presentan diferencias entre los modelos originales ResNet y FCN, y los modelos híbridos, ya que estos últimos omiten ampliamente la zona discriminativa que comprende toda la acción en la que el brazo vuelve a la posición inicial, solamente detectando como zona discriminativa la posición final. Así mismo se observa que mientras los modelos ResNet y FCN contienen regiones mucho más suaves representadas por los colores celeste, verde y amarilla, mientras que los modelos bilstm_resnet y bilstm_FCN contienen más subsecuencias de color rojo oscuro y azul que muestran que dichos modelos pueden filtrar las regiones no discriminatorias y discriminatorias con mayor confianza que sus pares originales.

En cuanto a dicha capacidad, reuce el modelo bilst_resnet sobre bilst_resnet, ya que se

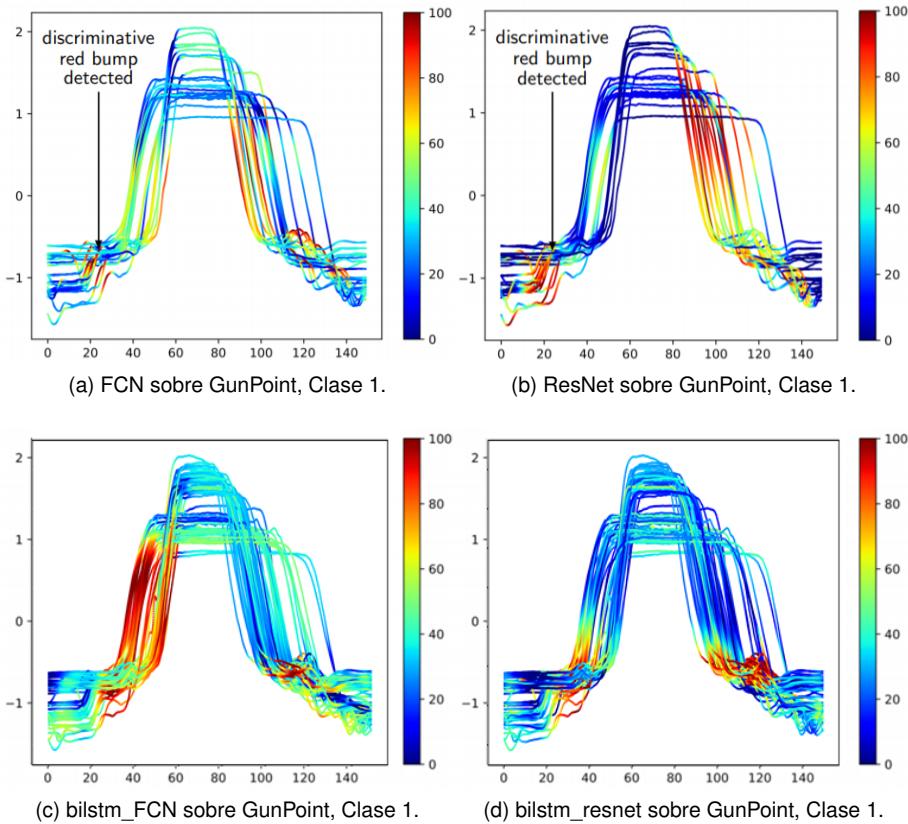


Figura 6.1: Mapa de activaciones sobre la clase 1 del conjunto de datos GunPoint para los modelos FCN, ResNet, bilstm_FCN y bilstm_resnet.
 Fuente: Fawaz et al. (2019), y elaboración propia, (2019).

denota un mayor proceso de filtrado de las regiones no discriminatorias y discriminatorias tanto en la clase 1 como la clase 2, lo que se debe a sus conexiones residuales entre las capas convolucionales que permite una extracción de características más detallada a través de los distintos filtros conjuntos.

Todas estas diferencias entre los modelos originales e híbridos, se deben a la introducción de la capa LSTM para la obtención de las dependencias temporales, anterior a las capas convolucionales de cada arquitectura. Esto supone la capacidad de las arquitecturas híbridas para manejar explícito del orden entre las observaciones cuando se localiza una forma discriminativa a través de la capa recurrente, y la capacidad para aprender características distorsionadas invariantes en el tiempo a través de las capas convolucionales. A partir de ello, se puede confirmar las inferencias teóricas iniciales, que indicaban que la capa LSTM era la encargada de extraer las dependencias temporales, mientras que las capas convolucionales realizaban la extracción de características. Sin embargo, se debe tener en cuenta que las conjeturas que se obtienen a partir de la observación de los mapas de activación de las figuras 6.1 y 6.2, no pueden confirmar qué modelo extrajo las subsecuencias más discriminativas, especialmente porque todos lograron un rendimiento similar en el conjunto de datos de GunPoint.

Finalmente se confirma experimentalmente la supremacía que presentan las arquitecturas híbridas de aprendizaje profundo para la clasificación de series de tiempo sobre los enfoques puramente convolucionales o recurrentes. Pese a ello, se debe señalar que en el estudio de dónde se obtienen los resultados de todos los modelos, las pruebas se hicieron 10 veces modificando la tasa de aprendizaje, a diferencia del presente estudio en el que se utiliza una validación cruzada de 5-fold.

Cabe destacar que no existen diferencias significativas en los tiempos de entrenamiento de bilstm_resnet y bilstm_FCN, con los modelos de aprendizaje profundo de la literatura, a pesar de que se complejiza la arquitectura con la adición de la capa recurrente.

6.3 COMPARACIONES CON OTROS MODELOS DE LA LITERATURA

Para las comparaciones con la literatura se incluyen los modelos de mayor rendimiento en la literatura como lo son COTE y HIVE-COTE, además de otros modelos con buen rendimiento como ST-HESCA, EE, TSF, TSBF, BOSS y LPS, y también un modelo ampliamente utilizado para tareas de TSC, como lo es DTW_F.

Se excluyeron modelos de aprendizaje profundo debido a que se comprobó experimentalmente que bilstm_resnet y bilstm_FCN presentan el mejor rendimiento, y aunque estos no poseen diferencias significativas con sus similares ResNet y FCN, se consideró que debido a que su

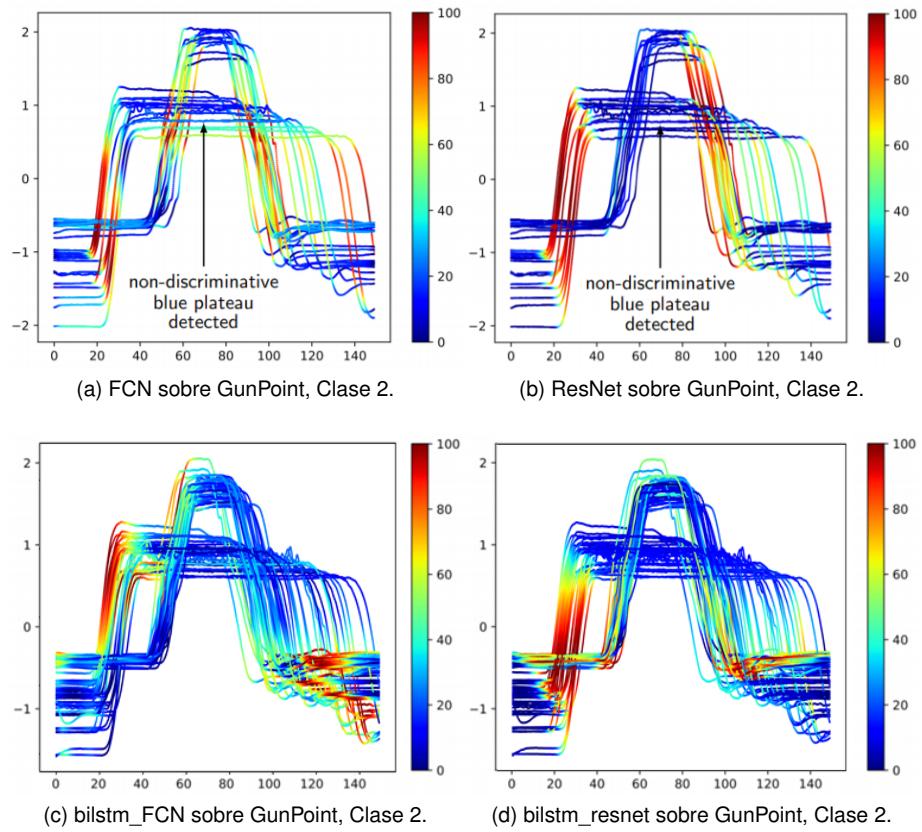


Figura 6.2: Mapa de activaciones sobre la clase 2 del conjunto de datos GunPoint para los modelos FCN, ResNet, bilstm_FCN y bilstm_resnet.

Fuente: Fawaz et al. (2019), y elaboración propia, (2019).

arquitectura es muy similar, bastaba sólo con considerar los modelos híbridos.

Tras las debidas experimentaciones con todos los conjuntos de datos bajo la métrica del accuracy, se obtiene que el modelo bilstm_resnet presenta 27 conjuntos en los que obtiene el mejor rendimiento, mientras que el modelo bilstm_FCN lo presenta en 24 ocasiones, siendo ambos los mejores bajo este parámetro. Además, examinando cada uno de los conjuntos de datos se detalla que en varios de ellos, ambos modelos presentan el mejor rendimiento muy por encima de los demás, como es el caso de ChlorineConcentration, Lightning2, SonyAIBORobotSurface1, SonyAIBORobotSurface2 y Yoga. Particularmente en lo que se refiere a los conjuntos de datos, se pudo establecer que los modelos propuestos alcanzan el mejor rendimiento en todos los dominios a excepción de MOTION, donde son superados por HIVE-COTE. A pesar de ello, lo más destacable es considerar que el ranking promedio del test de Friedman, muestra que bilstm_resnet en primer lugar y bilstm_FCN, superan a HIVE-COTE y COTE respectivamente, como los modelos de mayor rendimiento en tareas de TSC, lo que sin duda alguna, los convierte en modelos de vanguardia. Sin embargo, cabe destacar que no existen diferencias significativas entre estos modelos.

Continuando con el análisis comparativo, se debe señalar que los resultados base de los modelos de la literatura contemplan el uso de 100 remuestreos para el entrenamiento y prueba de los diversos modelos, mientras que los modelos híbridos aquí implementados solo son entrenados y probados con validación cruzada estratificada de 5 fold, por lo que si bien se presentan resultados de vanguardia, se debe seguir iterando sobre las pruebas de los modelos, para confirmar los resultados aquí implementados. Además se debe indicar que el presente estudio solo considera pruebas sobre 44 conjuntos de datos, en comparación a los 85 conjuntos disponibles en el repositorio UCR para TSC con los que se entran todos los modelos de la literatura. A pesar de ello, lo adelantado por el presente estudio puede sentar la base para comprobar que las arquitecturas híbridas de aprendizaje profundo pueden mejorar el rendimiento en los problemas de TSC.

Por otro lado, cabe destacar que COTE y HIVE-COTE, que presumían del mejor rendimiento hasta ahora, son clasificadores altamente complejos basados en conjuntos que ensamblan hasta 35 clasificadores distintos, por lo que su tiempo de entrenamiento es alto en comparación a los modelos de aprendizaje profundo que fácilmente pueden ser paralelizados.

Otro punto importante a destacar en el presente estudio es que a diferencia de HIVE-COTE, COTE, ST-HESCA, EE, BOSS, TSF, TSBF, LPS y DTW_F, los modelos aquí propuestos, no presentan un ajuste de los hiperparámetros para cada conjunto de datos, sino que se usa los mismos para toda la experimentación. Esto refleja el poder que presentan en general los modelos

de aprendizaje profundo para auto-ajustar su entrenamiento sean cual sean las condiciones y sin requerir mayor conocimiento de la problemática abordada. Sin embargo, si se llegasen a tener en futuros trabajos un menor rendimiento para los modelos híbridos aquí propuestos, una investigación más elaborada en la aplicación sobre cada conjunto de datos, según las características del dominio, la cantidad de clases, el número de instancias, el largo de la serie de tiempo, entre otras variables, puede mejorar el rendimiento de los modelos haciéndolos mucho más competitivos para afrontar problemas de TSC.

CAPÍTULO 7. CONCLUSIONES

Para concluir este trabajo de memoria se exponen a través de este capítulo las conclusiones obtenidas a partir de la evidencia y el análisis realizado a través de la investigación propuesta. Para esto el presente capítulo se dividen en tres partes: en la primera, se presentan las conjeturas finales que se pudieron recopilar en el desarrollo del presente trabajo, contrastando la hipótesis proveniente de la pregunta de investigación con los resultados obtenidos tras la experimentación. En la segunda parte se exponen los trabajos a futuro que pueden surgir de este trabajo, apuntando a proponer lineamiento de investigación que permitan seguir mejorando la clasificación de series de tiempo a través de arquitecturas híbridas de series de tiempo. Finalmente, se presentan las reflexiones personales del trabajo realizado, exponiendo las principales decisiones y caminos tomados a lo largo del proyecto de memoria para dar completitud a los objetivos.

7.1 CONJETURAS FINALES

La pregunta que dió pie a la investigación formulaba si una arquitectura híbrida de aprendizaje profundo sería capaz de mejorar la clasificación de series de tiempo con respecto a los métodos existentes, por lo que el objetivo general comprendía la implementación de un modelo híbrido de aprendizaje profundo basado en la combinación de redes neuronales recurrentes LSTM y redes convolucionales para trabajar con problemas de TSC. Una vez finalizado el estudio es posible concluir con los resultados expuestos que los modelos implementados no permiten verificar la hipótesis en su totalidad, porque si bien se obtuvieron resultados de vanguardia en general, es necesario profundizar el estudio en lo que se refiere a las iteraciones de la pruebas y en su aplicación a más conjuntos de datos.

Se implementaron en total 10 arquitecturas que se sometieron a prueba con 44 conjuntos de datos del repositorio para la clasificación de series de tiempo de la UCR. Los resultados bajo las métricas del coeficiente Kappa, el área bajo la curva ROC, el accuracy, la presición, puntaje F1, recall y el error logarítmico de entropía cruzada, determinaron en que los mejores modelos obedecieron a un enfoque lineal por sobre un enfoque por raíces, y una estructura una arquitectura en la que la primera capa es la recurrente antes de las posteriores capas convolucionales.

A través de los modelos empíricos en primera instancia no se lograron buenos resultados por lo que se tuvo que recurrir a implementar modelos híbridos basados en la literatura, para los cuales se tomo las arquitecturas ResNet y FCN propuestas en Wang et al. (2016), que presentaron un gran rendimiento en tareas de TSC. A partir de ello se esperaba que el rendimiento fuese

similar e incluso mejor, lo que en la práctica resultó en obtener los resultados de vanguardia aquí presentados.

La mejora en rendimiento se visualizó a través de los mapas de activación sobre el trabajo de los modelos con el conjunto de datos GunPoint, en donde se comprobó que el modelo bilstm_resnet alcanzaba una mayor capacidad para aprender a diferenciar las zonas discriminativas y no discriminativas en la serie de tiempo que incidían a la clasificación final. De esta forma se pudo dilucidar la capacidad de este modelo para aprender y detectar y determinar la importancia del orden entre las observaciones al localizar una forma discriminativa, y la capacidad para aprender características distorsionadas invariantes en el tiempo.

A través de ello se pudo confirmar las inferencias teóricas iniciales que contemplaban que la capa LSTM extraía las dependencias temporales, mientras las capas convolucionales procedían con la extracción de características.

En la experimentación se pudo mitigar el sobre-ajuste a través de técnicas como la normalización por lotes, la regularización L2, Dropout, MaxPooling y GlobalAveragePooling, aplicadas en diferentes arquitecturas, así como la técnica de la validación cruzada durante el entrenamiento. Además se hizo uso de la técnica de EarlyStopping con una espera de 200 épocas, buscando evitar el sobreajuste de los modelos, junto con verificar que estos generalizaran lo suficientemente bien y fueran eficientes en cuanto a las épocas de entrenamiento. En este proceso se hizo visible la necesidad de ajustar el parámetro de espera en ciertos conjuntos de datos que no generalizaban lo suficientemente bien en la cantidad de épocas a las que alcanzaban a llegar.

Se pudo enfrentar el desbalance de clases de muchos de los conjuntos de datos, a través de una validación cruzada estratificada, mitigando que los posibles resultados de precisión y error fuesen engañosos por la inclinación a clasificar siempre la clase dominante. A pesar de ello, se consideraron una multitud de métricas que pudiesen dar una mirada completa del rendimiento de los modelos, sabiendo en primera instancia la paradoja de la precisión, y los problemas que tiene el área bajo la curva con el desbalance de clases.

Finalmente, fue posible avalar empíricamente las ventajas que ofrecen los modelos de aprendizaje profundo en la clasificación de series de tiempo, ya que a través de estos, no se necesita realizar un proceso previo de extracción de características, puesto que este tipo de modelos diseñan sus propias características durante el entrenamiento. De esta forma, el investigador no requiere tener experiencia en el dominio de donde provienen los datos, lo que representa una gran ventaja en comparación con los otros enfoques estudiados.

7.2 LINEAMIENTOS FUTUROS

Como ya se venía adelantando, se plantea bajo los mismos modelos híbridos propuestos, profundizar el estudio en su aplicación a más de los 44 conjuntos de datos aquí dispuestos, con el fin de obtener mayores conjeturas acerca del rendimiento de estos en comparación a sus pares en la literatura.

En la misma linea, se propone profundizar la investigación ajustando los hiperparámetros de los modelos en forma general y en forma específica para cada conjunto de datos, asimilando el trabajo que proponen otros modelos de la literatura como lo hace COTE y BOSS.

Se propone explorar diversas técnicas de preprocessamiento de los datos, como la técnica de aumentación de datos, considerando aquellos conjuntos en los que se tienen pocas instancias para el entrenamiento de los modelos, así como explorar técnicas como Smote, para afrontar el desequilibrio de clases, diferente a la validación cruzada estratificada utilizada en el presente estudio.

A partir de los resultados es posible seguir generando diversos modelos con fundamentos en las arquitecturas ResNet y FCN, explotando un enfoque por raíces, e incluso un ensamble de ambas arquitecturas con la presencia de capas recurrentes, bajo el concepto de arquitecturas híbridas. Así mismo se propone seguir indagando en otros modelos que auguren resultados prometedores, tales como el modelo Encoder, no explotado en el presente trabajo y que presenta un gran rendimiento, sobresaliendo como el quinto entre los once modelos de aprendizaje profundo considerados.

Además se plantea implementar otro tipo de redes recurrentes como las unidades recurrentes cerradas GRU, o la implementación de capas convolucionales LSTM (ConvLSTM), similares a una capa LSTM, pero donde las transformaciones de entrada y las transformaciones recurrentes son convolucionales. En la misma linea, se plantea indagar en modelos generativos, y otros modelos no convolucionales.

El presente trabajo se sujeta solo a la experimentación con conjuntos de datos univariados, por lo que se propone implementar las variaciones pertinentes a los modelos propuestos para trabajar con conjuntos de datos de series de tiempo multivariadas.

De la misma forma, se plantea la implementación de las arquitecturas híbridas propuestas en otras tareas diferentes a las de TSC, tales como, agrupación, idexación y pronóstico, ya sea a nivel general o aplicado a ciertas problemáticas específicas. Y finalmente se propone indagar más técnicas de visualización de los modelos de aprendizaje profundo que permitan deslumbrar más luces del proceso de aprendizaje de los modelos y las reglas que se crean para realizar la clasificación.

7.3 REFLEXIONES FINALES

Finalmente se puede decir que se realizó un trabajo limpio y ordenado buscando mejorar la clasificación de series de tiempo bajo el enfoque seleccionado.

En el proceso se tuvieron varios errores por la inexperiencia en la aplicación de este tipo de investigaciones, principalmente en su diseño inicial, en donde no se determinaron oportunamente todas las pruebas necesarias para la configuración de los hiperparámetros de los modelos implementados, así como la aplicación de diversas técnicas que permiten mejorar el aprendizaje del modelo tales como la aplicación de diferentes inicializadores para los pesos de red, o el pre-procesamiento de los datos, lo que resultó finalmente en un retraso de los experimentos que redujo la brecha para tratar de implementar más modelos que pudiesen presentar mejores resultados.

A pesar de ello se presentan resultados robustos y concluyentes que dan pie para seguir indagando en la posibilidad que ofrecen los modelos de aprendizaje profundo para la clasificación de series de tiempo.

BIBLIOGRAFÍA

- Abanda, A., Mori, U., & Lozano, J. (2018). A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2), 378–412.
- Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *CoRR, abs/1302.6613*.
- Agrawi, A. (2017). Loss functions and optimization algorithms. demystified. Recuperado el viernes 30 de noviembre de 2018, de <https://medium.com/data-science-group-iitr/loss-functions-and-optimization-algorithms-demystified-bb92daff331c>.
- Albelwi, S., & Mahmod, A. (2017). A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19, 242.
- Alom, M. Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Essen, B. V., AbdulAwwal, & Asari, V. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *CoRR, abs/1803.01164*.
- Amr, T. (2012). Survey on time-series data classification. *TSDM 2012*.
- Avila, R., Torres, H., Gonzalez, L., Sánchez, L., Pérez, E., & Hidalgo, M. (2018). Deep learning, una revisión. *10.13140/RG.2.2.26893.84961*.
- Bagnall, A. (2014). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3), 565–592.
- Bagnall, A., Lines, J., Hills, J., & Bostrom, A. (2015). Time-series classification with cote: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9), 2522–2535.
- Bagnall, A. J., Bostrom, A., Large, J., & Lines, J. (2017). The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version. *Data Mining and Knowledge Discovery*, 31(3), 606–660.
- Bailly, A., Malinowski, S., Tavenard, R., Guyet, T., & Chapel, L. (2016). Bag-of-temporal-sift-words for time series classification. *AALTD'15 Proceedings of the 1st International Conference on Advanced Analytics and Learning on Temporal Data*, 1425, 9–15.
- Bakker, B. (2003). Reinforcement learning with long short-term memory. *NIPS'01 Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, (pp. 1475–1482).
- Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1), 1–127.
- Bre, F., & Gimenez, J. (2017). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. Recuperado el miércoles 28 de noviembre de 2018, de https://www.researchgate.net/publication/321259051_Prediction_of_wind_pressure_coefficients_on_building_surfaces_using_Artificial_Neural_Networks.
- Browniee, J. (2018). A gentle introduction to k-fold cross-validation. Recuperado el viernes 7 de diciembre de 2018, de <https://machinelearningmastery.com/k-fold-cross-validation/>.
- Browniee, J. (2019). Accelerate the training of deep neural networks with batch normalization. Recuperado el viernes 7 de diciembre de 2018, de <https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/>.

- Brownlee, J. (2018). Introduction to regularization to reduce overfitting of deep learning neural networks. Recuperado el miercoles 5 de diciembre de 2018, de <https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/>.
- Calvo, B., & Santafé, G. (2016). Scmamp: statistical comparison of multiple algorithms in multiple problems. *The R Journal*, 8(1).
- Colah, C. (2015). Understanding lstm networks. Recuperado el lunes 30 de abril de 2018, de <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. *CoRR*, *abs/1603.06995*.
- Dau, H. A., Bagnall, A. J., Kamgar, K., Yeh, C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., & Keogh, E. J. (2015). The ucr time series classification archive. *CoRR*, *abs/1810.07758*.
- Dauphin, Y., & Razvan Pascanu, K. C. S. G. Y. B., Caglar Gulcehre (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *CoRR*, *abs/1406.2572*.
- Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013a). A time series forest for classification and feature extraction. *Information Sciences*, 239, 142–153.
- Deng, H., Runger, G. C., Tuv, E., & Martyanov, V. (2013b). A time series forest for classification and feature extraction. *Information Sciences*, 239(1), 142–153.
- Deng, H., Runger, G. C., Tuv, E., & Martyanov, V. (2013c). A time series forest for classification and feature extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11), 2796–2802.
- Deng, L., & Yu, D. (2013). Deep learning methods and applications. *Foundations and Trends in Signal Processing*, 7, 197–387.
- der Vaart, A. V. (2010). *Time series*.
- Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., & Darrell, T. (2017). Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 677–691.
- Doukkali, F. (2017). Batch normalization in neural networks. Recuperado el viernes 7 de diciembre de 2018, de <https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c>.
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Recuperado el viernes 30 de noviembre de 2018, de <http://jmlr.org/papers/v12/duchi11a.html>.
- Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23 Issue 2, 419–429.
- Faust, O., Hagiwara, Y., Tan, J. H., Oh, S. L., & Acharya, R. (2018). Deep learning for healthcare applications based on physiological signals: A review. *Computer Methods and Programs in Biomedicine*, 161, 1–13.
- Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33, 917–963.
- García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10), 2044–2064.

- Geng, Y., & Luo, X. (2019). Cost-sensitive convolution based neural networks for imbalanced time-series classification. *Intelligent Data Analysis*, 23(2), 357–370.
- Haasdonk, B., & Bahlmann, C. (2004). Learning with distance substitution kernels. *DAGM-Symposium*.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Recuperado el miércoles 28 de noviembre de 2018, de <https://www.cs.toronto.edu/~hinton/science.pdf>.
- IBM (2017). El modelo de redes neuronales. Recuperado el lunes 30 de abril de 2018, de https://www.ibm.com/support/knowledgecenter/es/SS3RA7_18.0.0/modeler_mainhelp_client_ddita/components/neuralnet/neuralnet_model.html.
- Ihaka, R. (2005). *Time Series Analysis Lecture Notes for 475.726*.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.
- Jain, S. (2018). An overview of regularization techniques in deep learning. Recuperado el miércoles 5 de diciembre de 2018, de <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/>.
- Kang, E. (2016). Long short-term memory (lstm): Concept. Recuperado el miércoles 28 de noviembre de 2018, de <https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359>.
- Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2012). Character-aware neural language models. *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, (pp. 2741–2749).
- Kingma, D., & Ba, J. (2015). Adam: a method for stochastic optimization. *International Conference on Learning Representations*, (pp. 1–13).
- Korn, F., Jagaciish, H. V., & Faloutsos, C. (1997). Efficiently supporting ad hoc queries sequences in large datasets of time for systems. *ACM SIGMOD Record*, 26 Issue 2, 289–300.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *NATURE*, 521, 436–444.
- Lin, S., & Runger, G. (2018). Gcrnn: Group-constrained convolutional recurrent neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10), 4788–4797.
- Lines, J., & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3), 565–592.
- Lines, J., Taylor, S., & Bagnall, A. (2018). Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(52).
- Liu, C.-L., Wen-Hoar, & Tu, H. Y.-C. (2018). Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(6), 4788–4797.
- Mörchen, F. (2003). Feature-based classification of time-series data.
- Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (2001). Feature-based classification of time-series data. *Information processing and technology*, (pp. 49–61).
- Nielsen, M. A. (2016). Neural networks and deep learning. Recuperado el miércoles 28 de noviembre de 2018, de <http://neuralnetworksanddeeplearning.com/>.

- Parmar, A. (2018). Common loss functions in machine learning. Recuperado el viernes 30 de noviembre de 2018, de <https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23>.
- Popivanov, I., & Miller, R. J. (2002). Similarity search over time-series data using wavelets. *Proceedings 18th International Conference on Data Engineering*, 10.1109/ICDE.2002.994711.
- Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Recuperado el viernes 30 de noviembre de 2018, de <http://doi.org/10.1109/NNSP.1992.253713>.
- Ramirez, A. (2004). Metodología de la investigación científica. Recuperado el lunes 20 de agosto de 2018, de <http://www.postgradoune.edu.pe/pdf/documentos-academicos/ciencias-de-la-educacion/1.pdf>.
- Ratanamahatana, C. A., & Keogh, E. (2005). Three myths about dynamic time warping data mining. *Proceedings of the Fifth SIAM International Conference on Data Mining*, (pp. 506–510).
- Rouse, M. (2017). Aprendizaje profundo (deep learning). Recuperado el lunes 30 de abril de 2018, de <https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning>.
- Schäfer, P. (2015a). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29.
- Schäfer, P. (2015b). Scalable time series classification. *Data Mining and Knowledge Discovery*, 30.
- Serrà, J., Pascual, S., & Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. *CoRR, abs/1805.03908*.
- Settouti, N., El, M., Bechar, A., & Chikh, M. A. (2016). Statistical comparisons of the top 10 algorithms in data mining for classification task. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(1), 46–51.
- Sharma, A. (2017). Understanding activation functions in neural networks. Recuperado el viernes 30 de noviembre de 2018, de <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>.
- Shi, X., Chen, Z., Wang, H., & Yeung, D.-Y. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems*, 1, 802–810.
- Shumway, R., & Stoffer, D. (2000). *Time series analysis and its applications*. ISBN 978-1-4757-3261-0.
- Skymind (2017). A beginner's guide to recurrent networks and lstms. Recuperado el lunes 30 de abril de 2018, de <https://deeplearning4j.org/lstm.html>.
- Strodthoff, N., & Strodthoff, C. (2019). Detecting and interpreting myocardial infarctions using fully convolutional neural networks.
- Wang, X., Wirth, A., & Wang, L. (2007). Structure-based statistical features and multivariate time series clustering. *IEEE Internations Conference on Data Mining, Workshop on Knowledge Discovery and Data Mining from Multimedia Data and Multimedia Applications*, (pp. 351–360).
- Wang, Z., Yan, W., & Oates, T. (2016). Time series classification from scratch with deep neural networks: A strong baseline. *2017 International Joint Conference on Neural Networks (IJCNN)*, (pp. 1578–1585).
- Zeiler, M. (2012). Adadelta: An adaptive learning rate method. *CoRR, abs/1212.5701*.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. (2014). Time series classification using multi-channels deep convolutional neural networks. *International Conference on Web-Age Information Management WAIM 2014*, (pp. 298–310).

APÉNDICE A. RESULTADOS GENERALES PARA LOS MODELOS BAJO LA MÉTRICA DEL ERROR

A continuación se presentan los resultados generales para todos los modelos y los conjuntos de datos bajo la métrica del error.

Tabla A.1: Error promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.

Conjuntos de datos	lstm_cnn	cnn_bilstm	cnn_lstm_separated	multi_cnn_bilstm2	cnn_lstm_separated2	multi_cnn_bilstm	resnet_bilstm	bilstm_resnet	FCN_bilstm	bilstm_FCN
Adiac	1.1	1.23	0.87	4.86	0.84	1.24	1.22	0.58	1.08	0.84
Beef	0.71	0.65	1.38	1.97	1.08	0.93	0.39	0.78	0.34	0.64
CBF	0.0	0.0	2.73	0.03	2.88	0.01	0.0	0.0	0.0	0.0
ChlorineConc	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CinCECGtorso	0.0	0.0	0.01	0.01	0.01	0.01	0.04	0.0	0.02	0.01
Coffee	0.0	0.01	0.03	0.04	0.01	0.1	0.0	0.0	0.01	0.0
CricketX	1.43	1.23	2.46	1.9	1.75	2.07	1.02	0.43	1.0	0.51
CricketY	1.36	1.12	2.32	1.84	1.69	1.93	0.97	0.6	1.06	0.59
CricketZ	1.15	1.02	2.47	1.42	1.68	1.61	0.87	0.47	0.92	0.51
DiatomSizeR	0.0	0.07	0.02	0.28	0.01	0.07	0.01	0.0	0.11	0.0
ECGFiveDays	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FaceAll	0.13	0.09	0.15	0.07	0.16	0.08	0.02	0.01	0.03	0.03
FaceFour	0.55	0.51	1.33	0.88	0.96	0.49	0.0	0.02	0.01	0.0
FacesUCR	0.15	0.12	0.2	0.14	0.2	0.14	0.03	0.03	0.04	0.02
FiftyWords	1.85	1.86	3.0	2.81	2.34	3.16	1.31	0.65	1.46	0.62
Fish	0.57	0.65	0.63	0.87	0.58	0.74	0.14	0.08	0.23	0.1
GunPoint	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.01	0.0
Haptics	1.37	1.36	3.04	3.65	2.62	3.61	1.56	1.08	1.4	1.05
InlineSkate	1.61	1.54	2.54	4.83	2.49	13.73	2.58	1.41	1.86	1.1
ItalyPowerDemand	0.14	0.13	0.25	0.2	0.21	0.19	0.09	0.09	0.08	0.09
Lightning2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Lightning7	0.85	0.84	2.04	1.43	1.8	1.7	0.53	0.21	0.55	0.33
Mallat	0.05	0.01	0.01	0.13	0.01	11.23	0.0	0.0	0.01	0.0
MedicallImages	0.62	0.53	1.31	0.85	0.94	0.92	0.56	0.51	0.58	0.44
MoteStrain	0.12	0.11	0.16	0.16	0.17	0.17	0.07	0.05	0.11	0.06
NIInvFetalECGThx1	0.34	0.41	0.42	1.13	0.45	0.81	0.29	0.1	0.32	0.09
NIInvFetalECGThx2	0.31	0.43	0.37	0.78	0.38	0.55	0.23	0.13	0.27	0.16
OliveOil	0.3	0.39	0.44	1.06	0.48	0.81	1.2	2.22	1.06	1.3
OSULeaf	1.31	1.26	2.68	2.66	2.4	2.62	1.01	0.05	1.11	0.28
SonyAIBORobotS1	0.0	0.01	0.02	0.0	0.01	0.0	0.0	0.0	0.0	0.0
SonyAIBORobotS2	0.01	0.06	0.05	0.03	0.05	0.02	0.0	0.0	0.02	0.0
StarlightCurves	0.11	0.13	0.14	0.2	0.13	2.69	0.12	0.08	0.15	0.07
SwedishLeaf	0.43	0.49	0.52	0.72	0.48	0.41	0.1	0.12	0.13	0.1
Symbols	0.09	0.11	0.11	0.17	0.11	0.15	0.18	0.05	0.18	0.05
SyntheticControl	0.0	0.03	0.02	0.13	0.01	0.09	0.01	0.02	0.02	0.01
Trace	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0
TwoLeadECG	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TwoPatterns	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0
UWaveGestureLibX	0.49	0.54	0.92	1.42	0.78	1.41	0.76	0.49	0.68	0.46
UWaveGestureLibY	0.69	0.76	1.37	2.26	1.18	2.14	0.99	0.68	0.89	0.66
UWaveGestureLibZ	0.65	0.67	1.33	1.86	1.15	1.8	0.92	0.64	0.77	0.55
Wafer	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
WordSynonyms	1.32	1.36	2.42	2.59	1.96	2.41	1.19	0.55	1.41	0.67
Yoga	0.19	0.23	0.54	0.64	0.44	0.31	0.2	0.03	0.29	0.07
Ganados	15	10	8	7	8	9	15	27	10	28
Ranking Promedio	6.284 (4)	5.613 (6)	3.5000 (9)	2.886 (10)	4.227 (7)	3.590 (8)	6.715 (3)	8.102 (2)	5.840 (5)	8.238 (1)

Fuente: Elaboración propia, (2019).

Ranking de los modelos bajo la métrica del error:

1. bilstm_FCN
2. bilstm_resnet
3. resnet_bilstm
4. lstm_cnn
5. FCN_bilstm
6. cnn_bilstm
7. cnn_lstm_separated2
8. multi_cnn_bilstm
9. cnn_lstm_separated
10. multi_cnn_bilstm2

APÉNDICE B. RESULTADOS GENERALES PARA LOS MODELOS BAJO LA MÉTRICA DEL ACCURACY

A continuación se presentan los resultados generales para todos los modelos y los conjuntos de datos bajo la métrica del accuracy.

Tabla B.1: Accuracy promedio de la validación cruzada de cada uno de los modelos en cada uno de los conjuntos de datos.

Conjuntos de datos	lstm_cnn	cnn_bilstm	cnn_lstm_separated	multi_cnn_bilstm2	cnn_lstm_separated2	multi_cnn_bilstm	resnet_bilstm	bilstm_resnet	FCN_bilstm	bilstm_FCN
Adiac	0.74	0.73	0.8	0.24	0.78	0.7	0.75	0.86	0.78	0.83
Beef	0.69	0.78	0.52	0.68	0.69	0.76	0.83	0.79	0.89	0.83
CBF	1.0	1.0	0.92	0.99	0.92	1.0	1.0	1.0	1.0	1.0
ChlorineConc	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CinCECGtorso	1.0	1.0	1.0	1.0	1.0	1.0	0.99	1.0	1.0	1.0
Coffee	1.0	1.0	0.99	0.97	1.0	0.94	1.0	1.0	1.0	1.0
CricketX	0.6	0.65	0.6	0.68	0.6	0.7	0.74	0.85	0.73	0.84
CricketY	0.58	0.65	0.63	0.67	0.6	0.68	0.71	0.83	0.71	0.83
CricketZ	0.63	0.66	0.63	0.73	0.63	0.74	0.76	0.86	0.79	0.74
DiatomSizeRed	1.0	0.99	0.99	0.97	0.99	0.99	1.0	1.0	0.98	1.0
ECGFiveDays	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FaceAll	0.98	0.98	0.98	0.98	0.97	0.98	1.0	1.0	0.99	1.0
FaceFour	0.83	0.86	0.9	0.91	0.87	0.91	1.0	1.0	1.0	1.0
FacesUCR	0.98	0.97	0.97	0.98	0.96	0.98	0.99	1.0	0.99	1.0
FiftyWords	0.62	0.63	0.65	0.65	0.63	0.66	0.62	0.81	0.68	0.81
Fish	0.9	0.85	0.88	0.79	0.87	0.86	0.97	0.97	0.92	0.96
GunPoint	1.0	1.0	1.0	0.99	1.0	1.0	1.0	1.0	1.0	1.0
Haptics	0.46	0.5	0.45	0.46	0.46	0.46	0.37	0.48	0.42	0.32
InlineSkate	0.41	0.43	0.4	0.39	0.42	0.33	0.31	0.51	0.33	0.56
ItalyPowerDemand	0.96	0.96	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.97
Lightning2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Lightning7	0.71	0.78	0.63	0.72	0.62	0.7	0.84	0.91	0.84	0.88
Mallat	0.99	1.0	1.0	0.96	0.99	0.67	1.0	1.0	0.99	1.0
MedicallImages	0.78	0.82	0.78	0.81	0.79	0.83	0.8	0.84	0.82	0.85
MoteStrain	0.96	0.97	0.97	0.97	0.97	0.96	0.98	0.98	0.96	0.98
NInvFetalECGThx1	0.91	0.89	0.87	0.74	0.88	0.83	0.93	0.97	0.91	0.96
NInvFetalECGThx2	0.93	0.92	0.92	0.89	0.9	0.89	0.94	0.95	0.94	0.94
OliveOil	0.9	0.91	0.94	0.71	0.89	0.79	0.56	0.91	0.67	0.88
OSULeaf	0.51	0.56	0.58	0.59	0.54	0.62	0.75	0.99	0.71	0.97
SonyAIBORobotSurf1	1.0	0.99	0.99	1.0	0.99	1.0	1.0	1.0	1.0	1.0
SonyAIBORobotSurf2	0.99	0.98	0.99	0.99	0.99	0.99	1.0	1.0	1.0	1.0
StarlightCurves	0.97	0.97	0.96	0.96	0.92	0.91	0.97	0.97	0.95	0.98
SwedishLeaf	0.9	0.89	0.9	0.83	0.9	0.93	0.97	0.95	0.97	0.95
Symbols	0.98	0.97	0.97	0.97	0.97	0.98	0.94	0.99	0.95	0.99
SyntheticControl	1.0	0.99	0.99	0.97	0.99	0.98	0.99	0.99	0.99	0.99
Trace	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TwoLeadECG	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TwoPatterns	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
UWaveGestureLibX	0.83	0.82	0.82	0.79	0.82	0.82	0.82	0.82	0.79	0.83
UWaveGestureLibY	0.75	0.72	0.73	0.69	0.72	0.74	0.69	0.74	0.66	0.79
UWaveGestureLibZ	0.76	0.75	0.74	0.72	0.74	0.75	0.67	0.74	0.70	0.80
Wafer	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WordSynonyms	0.68	0.7	0.69	0.72	0.67	0.69	0.75	0.82	0.73	0.82
Yoga	0.96	0.93	0.91	0.89	0.9	0.92	0.94	0.98	0.94	0.98
Ganados	14	12	11	9	10	11	20	34	17	30
Ranking Promedio	5.807 (5)	5.931 (6)	6.557 (8)	7.170 (10)	6.955 (9)	6.261 (7)	4.886 (3)	3.057 (1)	5.182 (4)	3.193 (2)

Fuente: Elaboración propia, (2019).

Ranking de los modelos bajo la métrica del accuracy:

1. bilstm_resnet
2. bilstm_FCN
3. resnet_bilstm
4. FCN_bilstm
5. lstm_cnn
6. cnn_bilstm
7. multi_cnn_bilstm
8. cnn_lstm_separated
9. cnn_lstm_separated2
10. multi_cnn_bilstm2