Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes de Padel: Dominio Temporal versus Frecuencial

Autor: Claudia Martínez Valerio
Tutor: Daniel Gutiérrez Reina

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Ingeniería Electrónica, Robótica y Mecatrónica

Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes de Padel: Dominio Temporal versus Frecuencial

Autor:
Claudia Martínez Valerio

Tutor:
Daniel Gutiérrez Reina
Profesor titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2022

Proyecto Fin de Carrera: Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes de Padel: Dominio Temporal versus Frecuencial

Autor: Claudia Martínez Valerio

Tutor: Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

El primer y mayor agradecimiento está reservado para mi familia, en especial a mis padres y mi hermano, personas que me han apoyado y animado en cada instante de mi vida. También, a mis amigos y compañeros de clase, acompañándome en esas largas tardes de risas y agobios, prácticas y videollamadas, estudio y exámenes. Quiero dedicarles unas líneas a mis compañeras de piso, haciéndome los días de este último año más amenos y felices.

También querría agradecer a Daniel, por darme la oportunidad de formar parte de este proyecto de investigación y por ayudarme y estar pendiente en todo momento del progreso del mismo. Además, agradecer a Guillermo su trabajo previo y la atención recibida por su parte.

Para finalizar, una especial mención a todos los deportistas que han sacado una parte de su tiempo para hacer posible la realización de la base de datos: Ana, Ventura, Celia, Ana, Víctor, Pilar, Íker, Héctor, María, Sofía, Nacho y Christian.

Sin todas y cada una de estas personas, nada de esto hubiera sido posible.

Claudia Martínez Valerio
Sevilla, 2022

Resumen

El pádel es un deporte que está comenzando a tomar una gran relevancia en la sociedad actual. Sin embargo, existe una notable ausencia en la tecnología asociada a la monitorización y procesamiento de datos para esta actividad concreta.

El objetivo de este trabajo es clasificar golpes de padel a través de algoritmos de machine learning. Para ello, se confrontan los resultados obtenidos al realizar los entrenamientos, con la misma base de datos, en distintos dominios: temporal y frecuencial. Además, se comparan las consecuencias de añadir muestras de jugadores zurdos al conjunto de datos.

Como trabajo previo al entrenamiento de los algoritmos, se ha de crear la base de datos a utilizar, ya que no existe ninguna similar con la información adecuada.

Los algoritmos a estudiar son: redes neuronales, redes neuronales convolucionales 1D, redes neuronales convolucionales 2D, árbol de decisión, K vecinos más próximos y máquinas de vector soporte.

Abstract

Paddel is a sport that is beginning to take on great relevance in today's society. However, there is a notable absence in the technology associated with the monitoring and processing of data for this specific activity.

The objective of this study is to classify paddel strokes through machine learning algorithms. In order to, the results obtained when performing the training sessions are compared, with the same database, in different domains: temporal and frequency. In addition, the consequences of adding samples of left-handed players to the data set are compared.

Before training the algorithms, the database to be used must be created, since there is no library available with the appropriate information.

Algorithms to study are: neural networks, 1D convolutional neural networks, 2D convolutional neural networks, decision tree, K nearest neighbors and support vector machines.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

IMU	Inertial Measurement Unit
HAR	Human Activity Recognition
ANN	Artificial Neural Network
HMM	Hidden Markov Models
NB	Naive-Bayes
MQTT	Message Queing Telemetry Transport
LSTM	Long Short-Term Memory
WPT	World Padel Tour
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
m	metros
cm	centrímetros
g	gramos
GDL	Grados De Libertad
RAM	Random Access Memory
GB	gigabyte
IMU	Inertial Measurement Unit
GPIO	General Purpose Input/Output
V	Voltio
A	Amperio
mm	milímetros
Hz	Hercios
RNA	Red Neuronal Artificial
SVM	Support Vector Machines
KNN	K Nearest Neighbour

1. INTRODUCCIÓN

Los científicos investigan lo que ya es; los ingenieros crean lo que nunca ha sido.

-Albert Einstein -

La sociedad actual, en los últimos años, ha experimentado una creciente búsqueda de una vida más saludable, o comúnmente llamada "*vida fitness*". Esto implica un aumento en la práctica de deporte diario, así como el deseo de mejora en dicha actividad física [1].

La tecnología ha revolucionado este modo de vida a través de los *wearables*, dispositivos electrónicos, en los que va incorporado un microprocesador, que se colocan en alguna parte del cuerpo interactuando continuamente con el usuario, recogiendo datos, con el fin de realizar alguna actividad concreta. En el mundo deportivo, estos *wearables* son capaces de contar los kilómetros caminados en un día, de medir el ritmo cardiaco, de contar las calorías quemadas en un día, etc [2]. Para ilustrar el impacto real en la sociedad, en el año 2020 se llegaron a vender más de 444 millones de *wearables* [3], incrementando estas ventas en más de un 20% en el pasado año 2021, llegándose a vender más de 553 millones de estos [4].

¿Y si se aplicara esta recopilación de datos masiva al mundo del deporte? A esto se le conoce como *Big Data* y se puede afirmar que ha transformado el deporte de élite, ya que, expertos son capaces de perfeccionar la toma de decisiones basándose en la información recopilada sobre el campo [5]. El *Big Data* en el deporte va más allá, y su uso no se restringe exclusivamente a una toma justa de resoluciones sobre el terreno, ya que permite un mejor conocimiento del entorno competitivo (análisis del propio equipo y del rival) lo que posibilita la elaboración de una estrategia superior. Además, permite prevenir lesiones de los deportistas y, a la hora de fichar nuevos jugadores, seleccionar al que posea características que más se adecúen al equipo [6].

En un deporte como el pádel, se podrían proporcionar datos sobre los desplazamientos de los jugadores en el campo, zonas donde hay una mayor probabilidad de ganar un punto, qué tipos de golpes se dan en en las victorias y derrotas, cómo mejorar la técnica de golpeo, etc. El golpeo de la pelota juega un papel fundamental y el estudio de este durante entrenamientos o competición, puede ayudar al éxito deportivo. La clasificación y cuantización de los golpes realizados se podría realizar a través de un *wearable* que iría colocado en la muñeca del deportista con una IMU en su interior. Esta pasaría los datos a una red neuronal, ya entrenada, que realizaría la cuantización y clasificación [7].

Las señales captadas por la IMU se encuentran en el dominio del tiempo, pero ¿mejorarían la clasificación y cuantificación de los golpes si transformamos los datos al dominio de la frecuencia?, ¿qué ocurre al añadir movimientos de jugadores zurdos al conjunto de datos? Las respuestas a estas preguntas tienen gran importancia,

debido a la posibilidad de contribuir a la mejora del estudio realizado por Guillermo Cartes Domínguez [7].

En un gráfico en el dominio de la frecuencia se muestra qué partes de la señal están en cada banda de frecuencia dentro de un rango de frecuencias dado y sirve para realizar un análisis de la señal con respecto a la frecuencia [8]. Una de las ventajas que presenta el dominio de la frecuencia frente al del tiempo es la simplicidad de los cálculos matemáticos, ya que las ecuaciones diferenciales se convierten a ecuaciones algebráicas. Es por esto por lo que tiene sentido la primera pregunta realizada en el párrafo anterior.

1.1. Descripción del problema

Debido al reciente auge del padel en la sociedad actual existe una carencia de dispositivos electrónicos utilizados para la monitorización de esta actividad concreta: zonas de juego más frecuentadas, tipos de desplazamientos ejecutados por los jugadores, cuantificación y clasificación de los tipos de golpes realizados, etc. Los *wearables* actuales no disponen de estas funcionalidades.

Esta clasificación necesita un algoritmo matemático para clasificar los datos recibidos por los sensores, así como una base de datos representativa para entrenar dicho algoritmo.

1.2. Objetivos

Los objetivos principales del trabajo presentado son la comparación de algoritmos de *machine learning* para la clasificación de golpes de pádel con base de datos en el dominio de la frecuencia, así como el confrontamiento entre los diferentes resultados obtenidos entrenando la red neuronal con base de datos en el dominio del tiempo y en frecuencia. Además, se ampliará la base de datos disponible con el fin de mejorar el entrenamiento de la red y, con ello, los resultados. Por tanto, los objetivos se pueden resumir en:

- Ampliación de la base de datos disponible, a través de la introducción de datos nuevos.
- Modificación de la base de datos disponible transformando las señales del dominio del tiempo al de la frecuencia.
- Comparación diferentes métodos de aprendizaje automático (con base de datos en el dominio de la frecuencia) con el fin de obtener el mejor para la clasificación de golpes de pádel.
- Confrontación de resultados obtenidos al entrenar la red con la base de datos en el dominio del tiempo con los logrados al hacerlo con la base de datos en el dominio de la frecuencia con el objetivo de escoger el más efectivo.
- Visualización de consecuencias en los experimentos al añadir golpes realizados por deportistas zurdos al conjunto de datos.

2. ESTADO DEL ARTE

El fracaso es simplemente la oportunidad de empezar de nuevo, esta vez de forma más inteligente.

-Henry Ford-

En los últimos años, el reconocimiento de actividades humanas (HAR) ha tomado una gran relevancia debido a la creciente demanda por parte de los usuarios. Unos buscan, por un lado, la monitorización de su actividad física diaria, la contabilización de las calorías quemadas durante el día o tan solo la medición de su ritmo cardiaco. En cambio, los deportistas tanto profesionales como los que están en niveles inferiores buscan un incremento del rendimiento en el campo que practican.

El progreso de algunas técnicas de *machine learning* y el diseño de nuevos dispositivos con sensores integrados permiten el desarrollo de sistemas que satisfacen las necesidades de los consumidores, introduciendo nuevas funcionalidades en los *wearables* existentes.

Cada vez que se realiza un golpe en un deporte de raqueta como lo es el padel, la mano que sujeta la pala, realiza un movimiento distinto en función del golpeo que se haya ejecutado. Por tanto, resulta lógico considerar la diferenciación de golpes de padel como clasificación de movimientos de mano. Se pueden usar múltiples sensores disponibles en el mercado, aunque lo más común es medir las aceleraciones lineales y velocidades angulares, proporcionadas por la utilización conjunta de un acelerómetro con un giroscopio. En [9] se comenta la posibilidad de realizar la clasificación utilizando video análisis, lo que se descarta debido al coste económico y labor intensa de análisis.

En [10] Mohammed Mehedi Hassan, Md. Zia Uddin, Amr Mohammed y Ahmad Almogren pretenden desarrollar un reconocimiento de actividades humanas (HAR) robusto basado en los datos de los sensores disponibles en los teléfonos móviles, ya que son los dispositivos más utilizados por las personas. Proponen un diagrama de flujo compuesto por tres partes principales: sensores, extracción de características y clasificación. Los sensores de los *smartphones* utilizados son acelerómetros y giroscopios, capaces de proporcionar medidas entre 0 y 15 Hz. Comparan la capacidad de distintos algoritmos para identificar distintas actividades humanas como levantarse, sentarse, subir y bajar escaleras, hasta un total de 12. La red neuronal artificial (ANN) obtiene un accuracy general del 89.06%, la máquina vector soporte (SVM) un de 94.12% de aciertos y, por último, las redes de creencia profunda[1] (DBN) logra un accuracy general del 95.85%.

En [11] Manuel Gil-Martín, Rubén San-Segundo, Fernando Fernández-Martínez y Javier Ferreiros-López detectan el problema que, en la literatura, la mayoría de los estudios realizados para reconocer actividades humanas se basan en la división en ventanas superpuestas de las señales recogidas, cuando los movimientos normalmente duran más. Para solucionarlo, proponen el diseño de un HAR basado en 3 módulos principales:

[1] Las redes de creencia profunda pueden ser consideradas como una composición de redes simples y no supervisadas como las máquinas de Boltzmann restringidas

primeramente, dividen las señales en ventanas y extraen la información en frecuencia de cada una, a diferencia de los anteriores, que se realizaron en el dominio temporal. Seguido, clasifican la actividad utilizando redes neuronales convolucionales. Finalmente, aplican un postprocesado integrando la salida de cada ventana del módulo anterior en periodos más largos de tiempo. Las señales de aceleración fueron muestreadas a una frecuencia de 100 Hz, y se consideró que la actividad dura 5.12 s, por lo que el tamaño de la ventana fue de 512 muestras. Tras extraerse las características en frecuencia (aplicación de la FFT), el tamaño de la ventana se redujo a 256 muestras, con frecuencias desde 0 a 50 Hz. Las actividades humanas a frecuencias superiores de 25 Hz son ilegibles así que la longitud final de la ventana fue de 128 muestras. A continuación, introducen los datos en los algoritmos de redes convolucionales 2D y aplican el postprocesado, implentando un filtro de la mediana y técnicas basadas en el modelo oculto de Markov (HMM). Al aplicar el módulo de postprocesado obtienen un incremento del porcentaje de aciertos en las clasificaciones del 6.79% alcanzando un accuracy del 96.62%.

En [9] Joseph McGrath, Jonathon Neville, Tom Stewart y John Cronin realizan un estudio consistente en la clasificación de actividades de la parte superior del cuerpo en distintos terrenos y campos, utilizando IMUs y algoritmos de *machine learning*. Su labor se resume en la comparación de 20 artículos de investigación diferentes ya realizados, en los que se clasifican movimientos ejecutados en distintas actividades deportivas. Entre los deportes se encuentran 3 deportes de raqueta: tenis, bádminton y squash. Cada estudio realiza los experimentos con IMUs y algoritmos distintos. Llegan a la conclusión que no solo afecta en gran medida al porcentaje de clasificaciones correctas, el tipo de algoritmo escogido para entrenar al modelo, si no el tipo de IMU, sus especificaciones, el lugar donde se coloca, así como el tipo de preprocesado realizado a los datos, etc. Debido a la escasez de estudios relacionados no se puede escoger una combinación óptima para la clasificación.

Con respecto a la clasificación de golpes en un deporte de raqueta, en [12] Mingyue Wu, Mengjiao Fan, Yang Hu, Ran Wang, Yufan Wang, Yanchun Li, Shengyuan Wu y Guowen Xia con el fin de evaluar el nivel del jugador realizan una clasificación de 5 golpes de tenis gracias a la captación de señales de una IMU y su posterior tratamiento en algoritmos de *machine learning*. El experimento lo realizan 36 jugadores derechos divididos en 3 niveles amateur[2], medio y profesionales. Recogen un total de 5400 golpes, los cuales son preprocesados y segmentados como pasos previos a la extracción de características. A continuación, reducen las dimensiones de los datos y los introducen en los distintos clasificadores. Los algoritmos que comparan son las máquinas vector soporte, knn vecinos más cercanos y Naive-Bayes (NB). Las máquinas vector soporte, obtienen alrededor de un 85% de clasificaciones correctas, los knn de entorno al 46% y los algoritmos de NB alrededor del 64%.

El estudio anterior indica que la clasificación de golpes de raqueta utilizando una IMU y algoritmos de aprendizaje automático es posible.

En cuanto a pádel, un estudio a destacar es el realizado por Clara Menduiña Fernández en [13] que desarrolla un sistema capaz de obtener estadísticas en tiempo real de los deportistas durante un partido de pádel. Con esto, es posible deducir la estrategia de juego mientras se compite. Lo realiza a través de la sensorización de una pala de pádel con una IMU, y mediante la creación de una arquitectura MQTT de comunicación. Aplicando un filtro de Madgwick a las señales recogidas por los sensores se obtiene la rotación de la pala en los ejes x, y, z. A continuación, introduce los datos en una red neuronal recurrente de tipo LSTM. Para obtener las estadísticas de juego, los golpes que clasifica los guarda, con el fin de evaluar estadísticas. Por ejemplo, si se han realizado más globos que remates, se llega a la conclusión que se ha optado por una estrategia defensiva. El estudio tan solo pudo llevarse a cabo con 3 tipos de golpes: volea, bandeja y globo.

Otro estudio interesante en el mundo del padel es el llevado a cabo por Adrián Escudero-Tena, Bernardino Javier Sánchez-Alcaraz, Javier García-Rubio y Sergio J. Ibáñez en [14], donde se muestran nuevas aportaciones sobre indicadores de rendimiento en el pádel de élite. Recogen datos del WPT entre 2015 y 2019 de un total de 532 partidos y 1070 sets. La principal diferencia entre deportistas ganadores y perdedores resulta ser la efectividad del golpe siendo, las parejas ganadoras, las que presentan una mayor cantidad de *break points* o puntos de quiebre. Además, se mostró que las mujeres realizan una mayor cantidad de voleas en los partidos frente a la gran cantidad de remates realizados por los hombres, lo que supone un punto clave para los entrenadores, sabiendo dónde tienen que focalizar los entrenamientos.

[2] Jugadores que se están iniciando en el deporte

3. METODOLOGÍA

3.1. Datos en el dominio de la frecuencia Vs dominio del tiempo

En los objetivos de este trabajo, se ha hablado de entrenar las redes neuronales con las señales de la base de datos en el dominio de la frecuencia, pero ¿qué hay de diferencia con el dominio del tiempo?

Un gráfico en el dominio del tiempo muestra la evolución de una señal en el tiempo, analizando datos en función y durante un periodo de tiempo. Un ejemplo de las funciones que se examinan mediante el análisis en el dominio del tiempo son señales electrónicas, comportamientos de mercado y sistemas biológicos entre otras [23].

El dominio de la frecuencia analiza datos o funciones con respecto a la frecuencia, mostrando en sus gráficos, en función de la frecuencia en la que oscilan, las componentes de la señal [8]. Para convertir una señal del dominio del tiempo al de la frecuencia y viceversa, se debe de realizar una transformación. La transformación más común es la transformada de Fourier.

3.1.1. Transformada de Fourier

Las transformada de Fourier es una operación matemática sin la cuál, las telecomunicaciones modernas (internet, telefonía móvil, etc) no habrían podido desarrollarse más allá de una forma de comunicación local [24].

A pesar de que muchos matemáticos como Euler, Bernuilli, Lagrange o Gauss intervinieran en la invención de esta transformada fue Fourier quien se percató de que una función periódica se podía descomponer en la suma de funciones trigonométricas [24]. La transformación de Fourier generaliza este concepto obteniendo como resultado una suma de un número finito de ondas sinusoidales, recibiendo como entrada una señal con cualquier forma [23].

Una visualización de forma más sencilla en qué consiste la diferencia entre el dominio del tiempo y el dominio de la frecuencia, se puede observar en la Figura 1, donde en rojo se puede ver una señal en el dominio del tiempo, y en azul los 6 armónicos no nulos de la serie de Fourier, es decir, su representación en frecuencia.

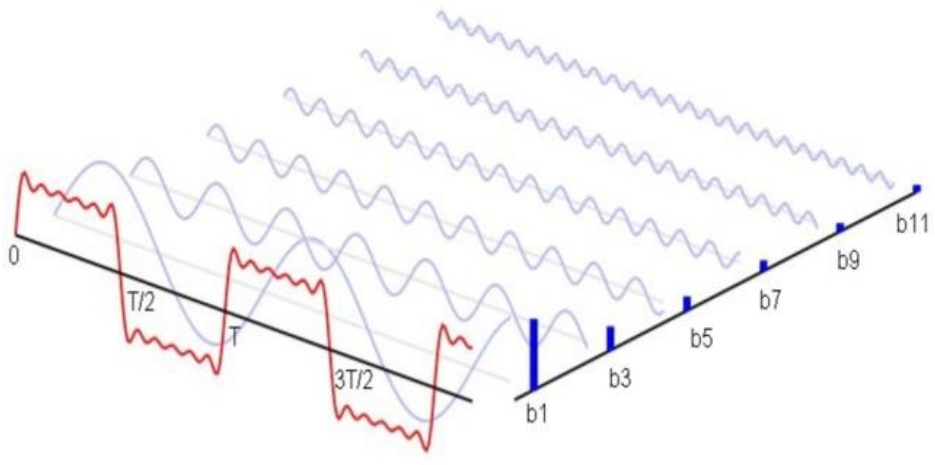


Figura 1. Señal en el dominio del tiempo vs en el dominio de la frecuencia. Disponible en [25].

3.1.2. Transformada Discreta de Fourier y Transformada Rápida de Fourier

3.1.2.1. DFT

La transformada discreta de Fourier o DFT es un método numérico utilizado para convertir una señal de entrada en el dominio del tiempo al dominio de la frecuencia. Requiere que la señal de entrada sea una secuencia discreta y de duración finita [26].

La DFT no debe de ser confundida con la transformada de Fourier, ya que la primera se obtiene muestreando la segunda [27].

3.1.2.1.1. Propiedades

Algunas de las propiedades destacadas son [28]:

- *Linealidad*: si una señal de entrada está multiplicada por un escalar, el resultado, es decir, la transformada, estará multiplicada por dicho escalar. Además, la suma de transformadas es igual a la transformada de la suma.
- *Dualidad*: si se vuelve a calcular la DFT a una expresión ya transformada se obtiene la señal original, invertida con respecto al eje vertical y escalada en N.
- *Convolución*: una convolución en el dominio del tiempo equivale a un producto en el dominio de la frecuencia. Por lo tanto, si se transforma una convolución es equivalente a realizar el producto de las transformadas.
- *Simetría*: con respecto al eje vertical

3.1.2.1.2. Cálculo

Se puede calcular a través de la siguiente expresión:

$$TDFx[n] = X[k] = \sum_{n=0}^{N-1} x[n] * e^{-\frac{j2\pi kn}{N}} \; ; k = 0, ; 1; \dots ; N-1 \qquad (1)$$

siendo:
x[n] → la secuencia que se desea transformar
N → el número de muestras
e → número racional, aproximadamente 2.71
X[k] → la transformada de la secuencia discreta de entrada

3.1.2.1.3. FFT

La transformada rápida de Fourier o FFT es una de las aplicaciones de la DFT, ya que es un algoritmo optimizado para la implementación de esta, reduciendo el tiempo de cálculo de n^2 pasos a $n*\log_2(n)$ [29]. Se aplica la FFT a una señal con el fin de facilitar información sobre la composición de la secuencia de entrada, descomponiendo sus componentes espectrales individuales. Dichos componentes tienen amplitud y fase determinadas cuya naturaleza son ondas sinusoidales simples a frecuencias discretas [30].

3.1.2.1.4. Funciones ventana

La FFT necesita que la señal de entrada sea un conjunto de datos finitos y discretos, lo que resulta un impedimento a la hora de aplicarla, ya que la mayoría de las señales son infinitas.

Un ejemplo para la explicación más clara con la finalidad de ilustrar los problemas de las señales infinitas y de la utilidad de las funciones ventana es la función seno simple. Su transformada debería de ser una barra vertical en la frecuencia del seno que se estudie (suponiéndola sin ruido). El resultado no es una única barra, si no varias. Esto ocurre porque la función intenta unir el primer y último valor, y aparecen discontinuidades, haciendo que se sumen frecuencias irreales [31].

La utilidad de las funciones ventana es evitar la aparición de las discontinuidades. Lo consiguen multiplicando el conjunto de valores, comenzando por los situados en la mitad y dirigiéndose hacia los extremos. Los números por los que se multiplican son progresivamente más pequeños, llegando a 0 en los extremos. De esta manera, cuando la función intente unir ambos extremos no existirá discontinuidad alguna.

3.1.3. Muestreo de una señal

Antes de continuar, conviene realizar una breve explicación del procedimiento de muestreo de una señal. Las señales del mundo real son la gran mayoría analógicas por lo que, si se quieren almacenar digitalmente, se necesitaría memoria infinita.

Es por esto la necesidad de simplificar la información, es decir, muestrear, tomando muestras durante intervalos regulares de tiempo y almacenándolas, representadas en la Figura 2 con puntos azules. Con esta información, si se reconstruye la señal uniendo las muestras almacenadas, se pierde información.

Si se cogen muestras durante intervalos más cortos, es decir, a mayor frecuencia, representados en la Figura 3 con puntos azules se capta con mayor detalle la señal.

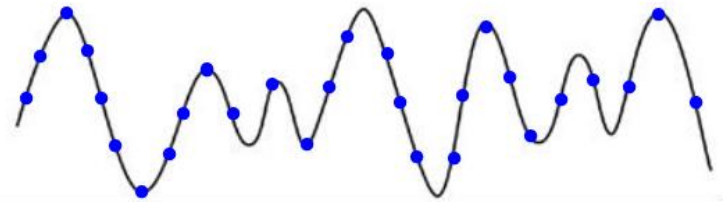


Figura 2. Señal analógica muestreada a frecuencia baja

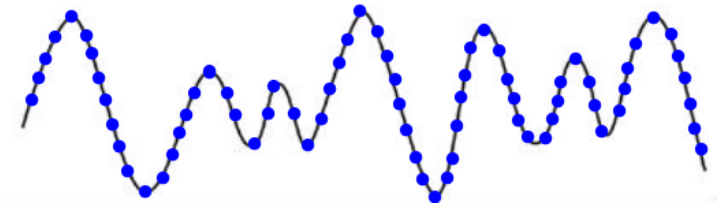


Figura 3. Señal analógica muestreada a frecuencia alta

Si la frecuencia de muestreo es muy pequeña con respecto a la de la señal muestreada, se pierde demasiada información. Es de esto de lo que va el teorema de muestreo de Nyquist-Shannon [32].

3.1.4. Teorema de muestreo de Nyquist-Shannon

El teorema afirma que, para poder reconstruir exactamente una señal muestreada, la frecuencia de muestreo debe de ser al menos el doble de grande que el ancho de banda, es decir, el doble de la frecuencia máxima a muestrear. Fue formulado por Nyquist en 1928 y demostrado por Shannon en 1949 [32], es decir, la señal analógica original queda totalmente descrita por la serie total de muestras obtenidas del muestreo.

Este teorema será fundamental para este trabajo. En los siguientes capítulos se verá el por qué.

3.1.4.1. Aliasing

¿Qué ocurre si no se cumple el teorema de Nyquist y se muestrea a una tasa inferior al doble del ancho de banda? Al realizar la interpolación de los puntos de muestreo, como se ha visto en el punto 3.3, de la señal original, se obtiene una señal totalmente distinta a la original. Esto se conoce como *aliasing* y no solo se da en el campo de la ingeniería si no que aparece también en la naturaleza: ruedas de coche cuando giran, aleteos de moscas, etc [33].

Una representación gráfica del efecto aliasing se puede observar en la Figura 4. Donde la señal azul es la señal analógica original, los puntos negros las muestras tomadas y la señal roja la reconstruida a partir de las muestras. Se puede comprobar que se ha reconstruido una señal totalmente diferente a la original.

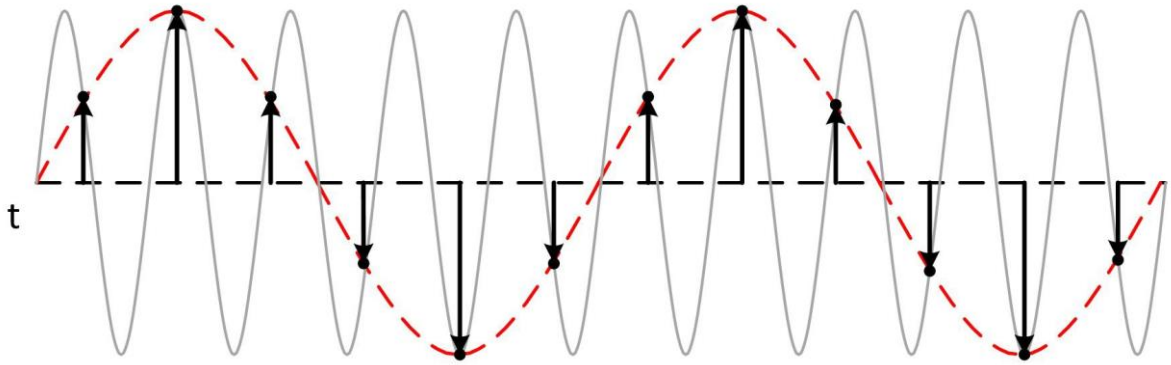


Figura 4. Aliasing. Disponible en [34].

3.2. Algoritmos de machine learning y deep learning

El concepto *Machine Learning* se utilizó por primera vez en 1959 y consiste en proporcionar a los computadores la habilidad de confeccionar predicciones e identificar patrones en conjuntos de datos copiosos. Es una parte del campo de la inteligencia artificial y sus técnicas son una parte esencial del *Big Data* [35].

Los algoritmos utilizados se pueden clasificar en tres:

- *Aprendizaje supervisado*: el aprendizaje supervisado entrena un algoritmno a través de un conjunto de datos previamente etiquetados con el fin de resolver problemas conocidos y realizar tareas específicas, es decir, los algoritmos aprenden de datos identificados y los aplican a entradas desconocidas para obtener la salida correcta [36]. Un ejemplo, podría ser la clasificación de perros y gatos a través de fotos, cuyas fotos de entrenamiento deberían de estar correctamente etiquetadas.
- *Aprendizaje no supervisado*: los datos utilizados no están etiquetados si no que se utilizan para detectar nuevos patrones por los algoritmos y detectar anomalías, pretendiendo dar sentido a dichos datos [36].
- *Aprendizaje por refuerzo*: el objetivo es que, mediante prueba y error y recompensando las decisiones acertadas, el modelo tenga la habilidad de tomar la mejor decisión frente a diversas situaciones, es decir, se busca que el algoritmo aprenda a través de la experiencia [35].

Puesto que se posee una base de datos etiquetados y el objetivo es clasificar nuevos datos de entrada en función de la clase a la que pertenezcan, se utilizarán y compararán varios algoritmos de aprendizaje supervisado. Será este tipo de aprendizaje al que se le aplicará un estudio más intensivo.

3.2.1. Conceptos básicos

Antes de profundizar en los algoritmos que se utilizarán para la clasificación, se han de introducir varios conceptos básicos. Se ha de mencionar, que a lo largo de las explicaciones se hará referencia al término modelo matemático. Esto es debido a que los algoritmos utilizados crean un modelo matemático que es el que realizará la clasificación, es decir, el encargado de identificar a qué clase pertenecen los datos de entrada. Aunque no todos los algoritmos presentan la necesidad de crearlo por la forma en la que se desarrollan.

3.2.1.1. Parámetros e hiperparámetros

Los parámetros constituyen las variables que permiten al modelo matemático realizar las predicciones, es decir, durante el proceso de entrenamiento el algoritmo, a través del conjunto de datos, se ajustan para realizar la clasificación correctamente. Una nota importante es que el modelo es el encargado de estimarlos, el programador no puede tocarlos [37].

En contraposición, los hiperparámetros son las variables que el programador debe ajustar y no dependen de los

datos. El procedimiento más habitual para su realización es mediante prueba y error, aunque suelen existir reglas genéricas con el fin de no probar valores sin sentido y tratar de optimizar el proceso [37].

3.2.1.2. Base de datos

Del conjunto total de datos disponibles, no todos son utilizados para el entrenamiento de los modelos, se pueden dividir en tres conjuntos:

Datos de entrenamiento: utilizados por el algoritmo para ajustar los parámetros del modelo matemático.
Datos de validación: encargados de validar el modelo entrenado, consiguiendo unos resultados de validación aceptables.
Datos de testeo: no intervienen en el entrenamiento del modelo en sí, si no que son los encargados de su evaluación final.

La cantidad de los datos totales utilizados para cada conjunto no es fija y se puede ajustar, aunque una buena división podría ser la mostrada en la Figura 5.

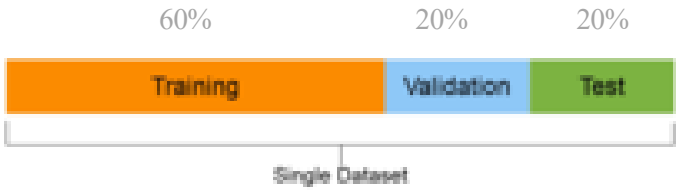


Figura 5. División del conjunto de datos totales. Disponible en [38].

3.2.1.3. Overfitting y regularización

El *overfitting* aparece cuando la base de datos con la que se entrena al modelo es demasiado amplia. El modelo aprende de los detalles y del ruido de los datos integrándolos como conceptos. El modelo pierde la capacidad de generalizar afectando negativamente al resultado [39]. Una posible solución es la regularización, existiendo distintas alternativas para su implementación.

3.2.2. Algoritmos para la clasificación

En esta sección se explicarán los distintos algoritmos utilizados para la clasificación de los golpes de padel en este trabajo.

3.2.2.1. Redes neuronales densamente conectadas

El modelo matemático de una Red Neuronal Artificial pretende asemejarse a la conducta que adoptan las neuronas en los organismos y cómo se comunican constituyendo el cerebro, identificándose este con la CPU [40].

Las computadoras actuales son muy rápidas, aunque el hecho de desempeñar tareas como el reconocimiento y la clasificación de patrones suponen un gran trabajo ralentizando sus labores. Al cerebro humano apenas le cuesta trabajo realizar dichas tareas por lo que se pensó en realizar modelos que se parecieran al funcionamiento de este. Las RNAs son un intento de recrear la manera en que el cerebro responde ante los estímulos del exterior [40].

La inteligencia conseguida por la red neuronal se consigue gracias al número de conexiones entre neuronas que, siguiendo con la analogía del cerebro, es en la sinapsis donde ocurren las reacciones bioquímicas que permiten conectarse a dos neuronas [41].

La estructura de una red neuronal se compone al menos una capa, aunque es habitual que existan varias. Las del medio, llamadas capas ocultas, es donde la información que se quiere clasificar es pasada de unas a otras no pudiéndose ver cómo se tratan los datos recibidos por cada una. Si existen varias capas, las únicas no ocultas son la de entrada y la de salida logrando ver los datos iniciales y los finales [42]. En las redes neuronales densamente conectadas, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente.

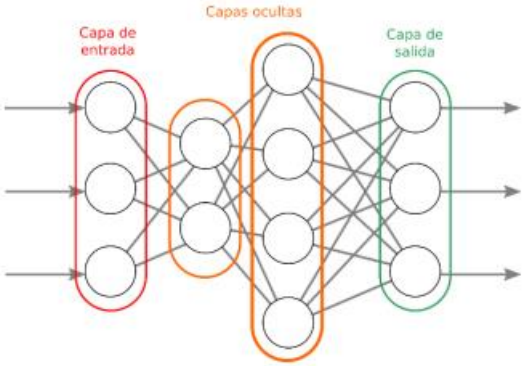


Figura 6. Esquema del modelo de una RNA. Disponible en [42].

3.2.2.1.1. Modelo de neurona de McCulloch-Pitts

Publicado en 1943, es considerado el asentamiento de las bases de las RNAs actuales. Consistió en examinar la capacidad de procesar y computar información que podía llegar a poseer el modelo matemático de una neurona [43].

El modelo inicial neuronal propuesto consistiría en canales de entrada y tan solo uno de salida. Las neuronas se podían conectar unas con otras existiendo dos tipos de conexiones: *excitadoras* si la neurona destino resulta excitada sumándole un 1 e *inhibidoras*, si contienen a la misma restándole un 1. La neurona destino puede resultar activada, si la suma (x) de todas las excitaciones e inhibiciones supera un umbral prefijado (U), estableciendo el valor de la salida a 1. En términos matemáticos se está aplicando la función escalón de *Heasivide* [43].

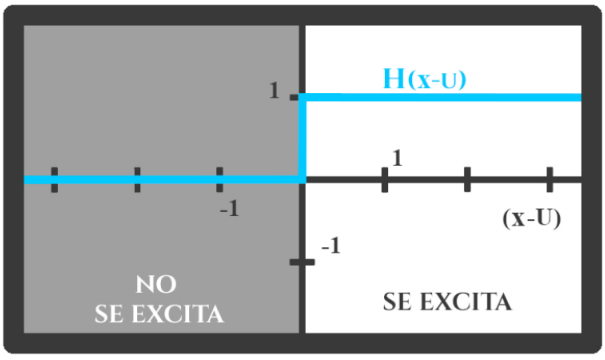


Figura 7. Función escalón de *Heasivide*. Disponible en [43].

Se puede llegar a la conclusión, de que el modelo matemático de la neurona de McCulloch-Pitts consta de dos operaciones:

- Cómputo de la suma de excitaciones e inhibiciones recibidas (x)

$$x = \sum_i c_i * d_i \tag{2}$$

  Siendo *c* una constante que puede tomar el valor de 1, si la neurona precedente es excitadora, o de -1, en el caso de que fuera inhibidora y *d*, otra constante con un valor de 0 si la neurona antecedente no ha sido activada y 1 en el caso que sí lo haya sido.

- Función escalón de *Heasivide*,

$$s = H(x - U) \tag{3}$$

  que determina si la neurona dispara o no, es decir, si se activa, en cuyo caso, *s*, valdría 1. En caso contrario pasaría a valer 0 [43].

3.2.2.1.2. Perceptrón simple

El perceptrón simple está inspirado en el modelo de neurona de McCulloch-Pitts y consta de una neurona básica cuyo objetivo es separar elementos etiquetados binarios en un plano [44].

Sus componentes se pueden dividir en 5 [45]:

- *Entradas (x)*: constituyen los datos que se desea clasificar.
- *Pesos (w)*: son los valores que se adquieren en el entrenamiento previo. Con los errores los valores de los pesos se adecúan y se actualizan.
- *Suma ponderada*: es el resultado de la suma total de cada entrada multiplicada por su peso correspondiente.

$$y = \sum_i x_i * w_i \tag{4}$$

- *Función de activación*: hay distintos tipos y cada una realiza una operación matemática distinta sobre el valor que se recibe. En apartados posteriores se estudiarán las más importantes.
- *Salida*: la suma ponderada se recibe en la función de activación y tras la operación correspondiente se obtiene la salida.

Los distintos componentes y su estructura se encuentran disponibles en la Figura 8.

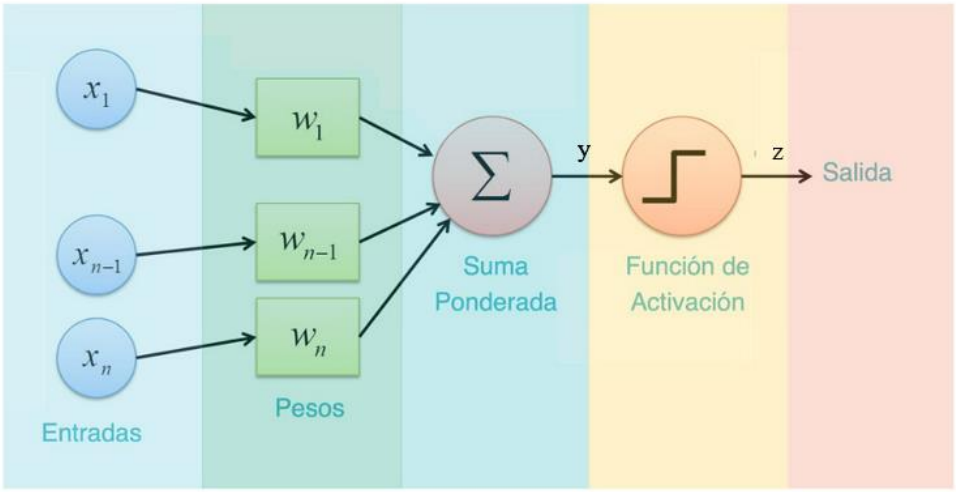


Figura 8. Estructura perceptrón. Disponible en [45].

En el caso actual, perceptrón simple, se trata de decidir si la característica de entrada pertenece a una clase o a otra, por lo que la salida será binaria:

$$z(x) = \begin{cases} 1, & Y < 0 \\ 0, & Y \geq 0 \end{cases} \tag{5}$$

siendo Y la recta que separará las regiones a clasificar, en este caso dos (si fueran tres sería un plano) y que viene dada por la siguiente ecuación:

$$Y = y + \theta \tag{6}$$

donde θ se corresponde a un sesgo o umbral y la y es la expuesta en la ecuación de la suma ponderada.

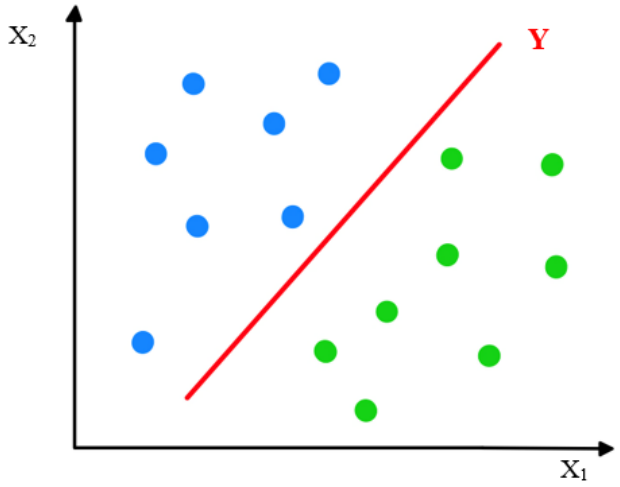


Figura 9. Ejemplo de perceptrón simple

Se puede concluir, que el entrenamiento del perceptrón simple se resume en determinar el umbral y ajustar los pesos sinápticos que hagan que la entrada se ajuste a la salida de una mejor forma. Se comienzan asignando valores aleatorios y se van corrigiendo según la diferencia entre los valores calculados por el perceptrón y los deseados, conformando un proceso iterativo [46].

El perceptrón simple tan solo es capaz de realizar una clasificación binaria, ya que para separar un mayor número de clases se necesitarían más neuronas, convirtiéndose en un perceptrón multicapa.

3.2.2.1.3. Función de activación

En el apartado anterior se ha mencionado el concepto de función de activación que, en resumen, es la operación matemática que determina la salida de la neurona permitiendo introducir no linealidades en el modelado de la red. Las funciones de activación más comunes son [47]:

- *Linear*: la salida es igual a la entrada

$$f(x) = x \tag{7}$$

- *Sigmoid*: reduce valores extremos transformándolos a una escala de 0 a 1 donde los valores altos tienden a 1 y los bajos a 0

$$f(x) = \frac{1}{(1 - e^{x})} \tag{8}$$

- *Tanh (tangente hiperbólica)*: es similar a la *sigmoid*, reduce valores extremos transformándolos a una escala de -1 a 1 donde los valores altos tienden a 1 y los bajos a -1

$$f(x) = \frac{2}{(1 + e^{-2x})} - 1 \tag{9}$$

- *ReLU*: los valores superiores a un umbral establecido permanecen invariantes, en cambio anula a los inferiores.

$$f(x) = max(0, x) = f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{10}$$

- *Softmax*: convierte las entradas a una representación en forma de probabilidades. Es característica de la última capa de un clasificador multiclase, ya que garantiza que el sumatorio de todas las salidas sea igual a 1, escogiendo como predicción final la probabilidad más alta.

$$f(x)_i = \frac{e^{x_i}}{\sum e^{x_i}} \tag{11}$$

3.2.2.1.4. Perceptrón multicapa

Anteriormente, se ha mencionado que para realizar una clasificación no binaria el perceptrón simple compuesto

por una neurona no era de utilidad por sí solo, si no que se requería de su conexión con más neuronas. Dichas neuronas conectadas, pueden agruparse de tal manera que la salida de unas se convierte en la entrada de otras formando capas. Si cada neurona de una capa se conecta con todas las neuronas de la capa siguiente, se ha formado una red *neuronal densamente conectada*, también llamado perceptrón multicapa [45].

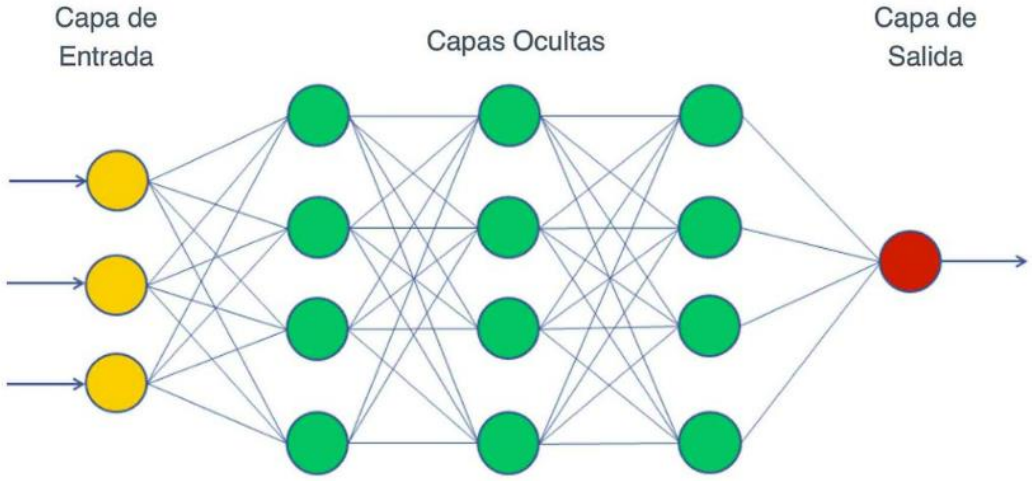


Figura 10. Red neuronal densamente conectada. Disponible en [45].

3.2.2.1.5. Proceso de entrenamiento

Una RNA densamente conectada está formada por neuronas unidas entre ellas a través de conexiones con pesos asignados, dictaminando la importancia que tendrá esa unión en la neurona al multiplicarse por el valor de entrada. El entrenamiento consiste en ajustar los pesos (w_{ij}) y sesgos (b_i) con el fin de hacer que la entrada se ajuste a la salida de la mejor forma posible.

El proceso de ajuste es iterativo y se puede dividir en fases [48]:

- *Forwardpropagation*: primera fase en la que los datos cruzan la red neuronal desde la entrada hasta la salida pasando por todas las capas para que sus predicciones (*labels*) sean calculadas. Todas las neuronas aplicarán su transformación a la información que le pasen las neuronas de la capa anterior y se la pasará a las neuronas de la capa siguiente. Una vez los datos hayan atravesado toda la red, la última capa tendrá una predicción. El siguiente paso es utilizar la *loss function*.
- *Loss function*: segunda fase en la que se mide cómo de buena ha sido la predicción en relación con el resultado correcto, estimando de esta forma el error cometido. Se puede comparar con lo correcto ya que se está entrenando con datos etiquetados correctamente. Se busca que el error sea cero por lo que el modelo irá ajustando los pesos de manera automática hasta conseguir buenos resultados. Cuando se ha calculado la *los* se pasa a la siguiente fase.
- *Backpropagation*: en esta tercera fase, la información *loss* se propaga desde la capa de salida hacia la capa de entrada. Un aspecto a tener en cuenta es que las neuronas no reciben toda la información que se transmite por la red, si no una fracción de la señal total. Dicha fracción se calcula aproximadamente con la contribución que haya aportado la neurona a la salida inicial. Cuando se haya llegado otra vez a la capa de entrada se aplicará la técnica *gradient descent*.
- *Gradient descent*: el objetivo principal es minimizar la *loss*, por lo que, cuando dicha información haya llegado a la capa de entrada, se calculará su derivada y con ella se ajustarán los pesos a través de pequeños incrementos. Esto permitirá ver en qué dirección desciende el mínimo global.

  Un esquema visual de las etapas está disponible en la Figura 11.

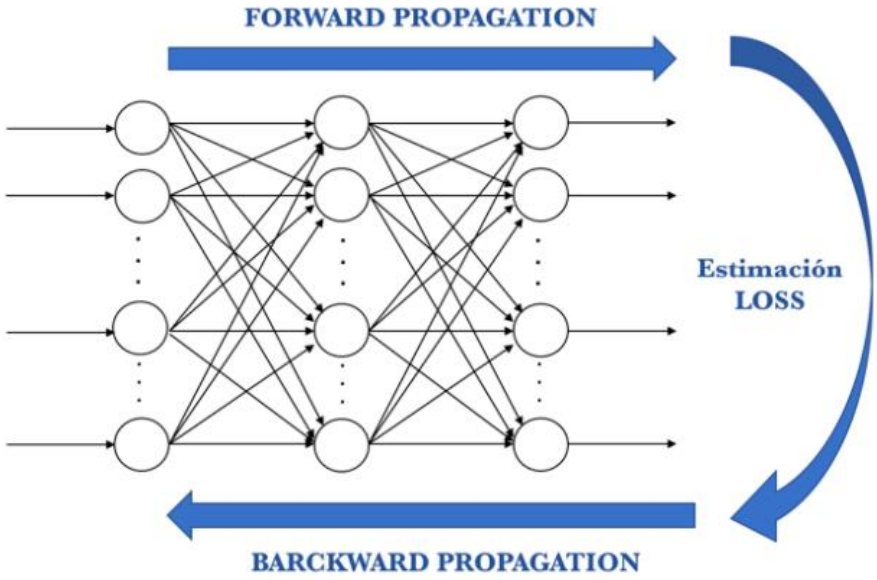


Figura 11. *Forward propagation, Loss function* y *Backward propagation*. Disponible en [48].

Estas fases se repiten iterativamente el número de veces indicado en el hiperparámetro *epochs*, es decir, el número de *epochs* es la cantidad de veces que pasan los datos por la red. Dicha información se divide en lotes de datos (*batches*). Por tanto, el *batch size* es el hiperparámetro que expresa el tamaño del lote que pasará por la red en una iteración del entrenamiento. El último hiperparámetro más característico de este tipo de algoritmos es el *learning rate*, es la rapidez con la que el modelo ajusta los pesos en el *gradient descent*. Si es demasiado grande se darán pasos gigantes agilizando el proceso de aprendizaje, aunque es probable que se salte el mínimo y se obtenga un efecto indeseado. Se puede ver reflejado este efecto en la Figura 12, donde el mínimo está representado con la flecha azul.

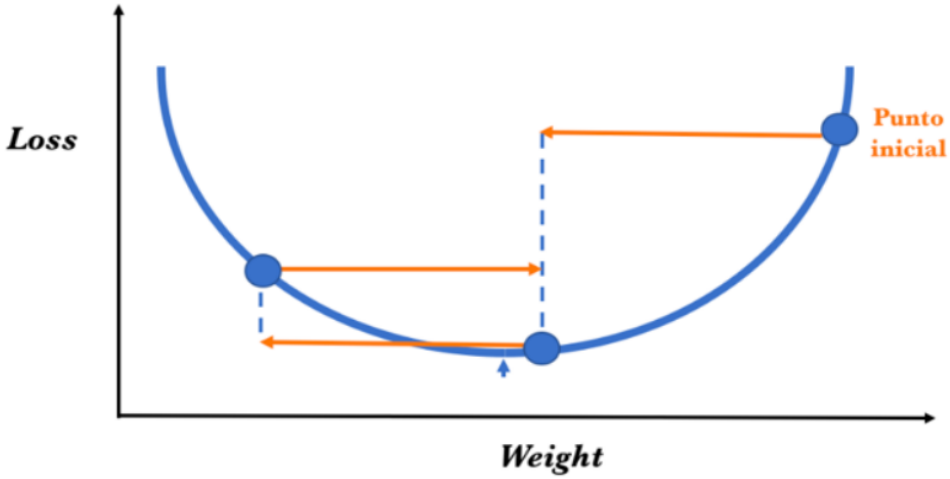


Figura 12. Efecto indeseado del *learning rate* demasiado grande. Disponible en [48].

3.2.2.1.6. Overfitting y regularización

En el apartado 3.2.1.3, se ha explicado el concepto de *overfitting* y la importancia de evitarlo. En las redes neuronales densamente conectadas, un método de regularización se puede introducir a través de una capa *dropout*, que desactivará arbitrariamente neuronas durante el entrenamiento. De esta forma el modelo no se hará dependiente de un grupo pequeño de neuronas y se realizará un aprendizaje más rápido ya que se evita el tener que dar valor a los pesos de dichas neuronas [49].

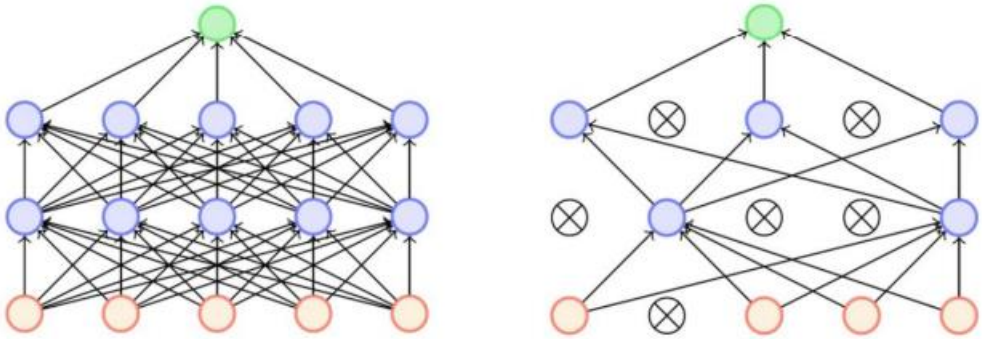


Figura 13. Red neuronal densa con *dropout*. Disponible en [49].

3.2.2.1.7. Keras

Keras se trata de una librería escrita en Python de código abierto utilizada para crear e implementar redes neuronales y es la que se utilizará para llevar a cabo este trabajo [50].

3.2.2.2. Redes convolucionales 1D

Una red neuronal convolucional es un caso concreto de red neuronal, similar a la del apartado anterior, cuyas neuronas tienen parámetros aprendidos (sesgos y pesos). Las capas convolucionales se caracterizan por aprender en ventanas de dos dimensiones de reducido tamaño patrones locales, a diferencia de las capas densas ya estudiadas, las cuales aprenden patrones globales en el espacio total de los datos de entrada [48].

Otro punto a destacar de las capas convolucionales es su capacidad de aprender jerarquizando espacialmente los patrones, es decir, a medida que se adentra más profundamente en la red las capas aprenden patrones cada vez más complejos. Por ejemplo, una primera capa aprendería patrones básicos y los aprendidos por la siguiente un poco más complejos formando pequeños conjuntos de los básicos aprendidos en la capa anterior [48].

Las conexiones de las pequeñas zonas localizadas se realizan a través de un filtro o *kernel*, cuya longitud determina el número de datos que formarán la pequeña zona localizada a estudiar. La salida de dichos datos será un único valor y se obtendrá a través de la aplicación del producto escalar a los datos de la subsecuencia de entrada y un vector *kernel* de la misma longitud de pesos aprendidos. La máscara *kernel* va recorriendo la secuencia completa de entrada [51].

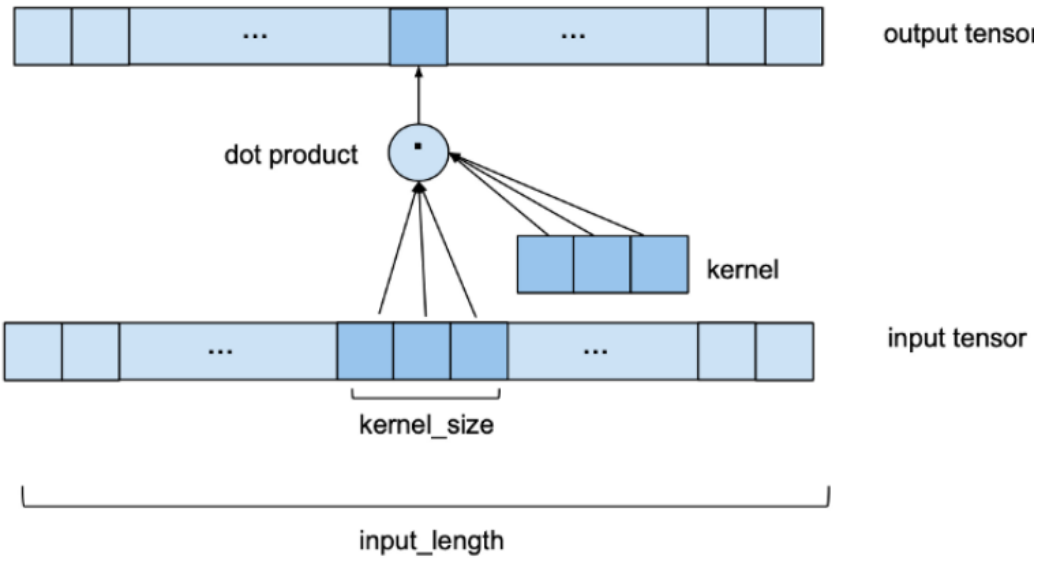


Figura 14. Convolución con *kernel*. Disponible en [51].

No se aplica solo 1 kernel, si no que existirán varios. A este conjunto se le denomina filtros. Por ejemplo, si a

una imagen de 30x30 píxeles se aplica una convolución con 50 filtros, se obtendrán a la salida de la capa un total de 50 matrices de dimensiones 30x30. Para esta operación, se tendrá una primera capa oculta de la red con un total de 45000 neuronas [52]. Si se aumentan tanto los píxeles de la imagen como las capas se obtienen una cantidad desorbitada de neuronas. Es por este motivo, la necesidad de muestreo conservando las características más importantes recogidas por cada filtro.

3.2.2.2.1. Operación de Pooling

El objetivo principal de las capas de *pooling*, es simplificar la información obtenida por las capas convolucionales y crear una versión condensada. Consiguen reducir el número de parámetros a aprender y la cantidad de información que computar. Es por esto, por lo que las capas de *pooling* se sitúan inmediatamente después de las convolucionales. Existen varios tipos de capas de *pooling* [53]:

Max Pool: selecciona el elemento máximo de la región a estudiar.
Average Pooling: la salida será la media de los elementos de la región estudiada.

En la Figura 15, se expresa de una forma visual los tipos de capas de *pooling* explicados.

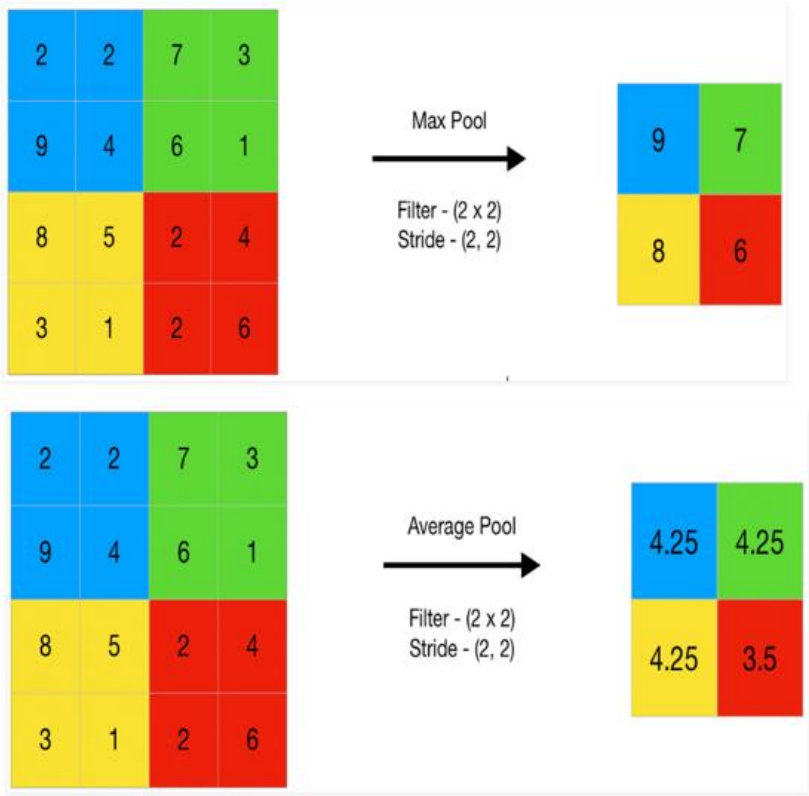


Figura 15. Tipos de capas *pooling*. Disponible en [53].

Hiperparámetros En Keras, los principales hiperparámetros de una red convolucional 1D son [48]:

- *Tamaño y número de filtros*: el tamaño representa el tamaño de la ventana de convolución que recorre la serie de entrada y, el número de filtros, representa el número de características que se empleará para realizar la clasificación.
- *Padding*: el objetivo es añadir ceros en los extremos del conjunto de datos de entrada como paso previo al paso de la ventana, de tal forma que se obtenga una imagen de salida de las mismas dimensiones que la de entrada.
- *Stride*: indica el número de pasos en los que se mueve la ventana del *kernel*. A mayor *stride*, menor información se pasa a la siguiente capa.
- *Función de activación*: vista anteriormente e igualmente aplicable a las redes neuronales convolucionales.

3.2.2.3. Redes neuronales convolucionales 2D

Las redes neuronales convolucionales 2D siguen siendo redes convolucionales por lo que los principios de funcionamiento son los mismos. La diferencia es la dimensionalidad de los datos de entrada y cómo el kernel los recorre.

En la Figura 16, la imagen de la izquierda representa cómo el filtro recorre los datos desplazándose en una única dirección. En cambio, el control deslizante realizado por el kernel en la red convolucional 2D de la derecha, es capaz de desplazarse en dos direcciones.

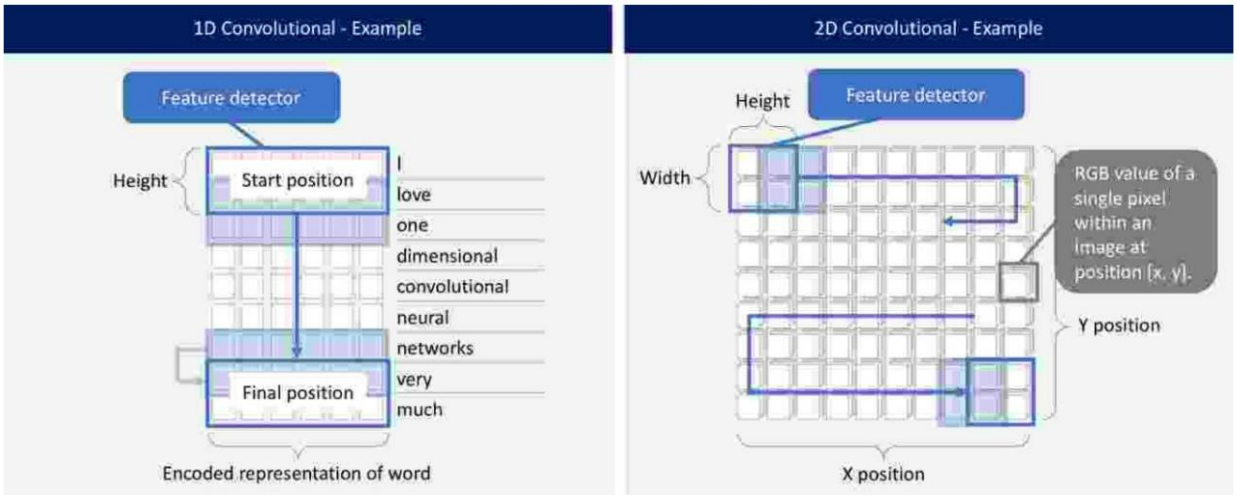


Figura 16. Red convolucional 1D vs red convolucional 2D. Disponible en [54].

3.2.2.4. Árbol de decisión

El árbol de decisión (AD) es un algoritmo de aprendizaje automático en el que se clasifican datos en distintas clases a través de la creación de una estructura tipo árbol. El árbol contiene condiciones anidadas *if-else* de tal forma que un objeto que entre en el árbol irá contestando las distintas preguntas que se le planteen hasta lograr clasificar su clase al final [55].

Se componen de distintos nodos que serán los encargados de formular las preguntas [55]:

- *Nodo decisión*: representa una característica de la base de datos y de estos saldrán tantas ramas como posibilidades tenga dicha característica. Por ejemplo, si el nodo representa el color y se están clasificando fichas de parchís, dicho nodo tendrá 4 posibles salidas: una para el rojo, otra para el azul, otra para el verde y otra para el amarillo. Las salidas de estos nodos se conectan a otro nodo de decisión o a un nodo hoja:
- *Nodo hoja*: son los nodos finales y representan la clase del objeto clasificado por lo que tan solo tendrán una salida.
- *Nodo raíz*: es un tipo de nodo de decisión y representa el conjunto total de los datos disponibles, por lo que será el inicio del árbol.

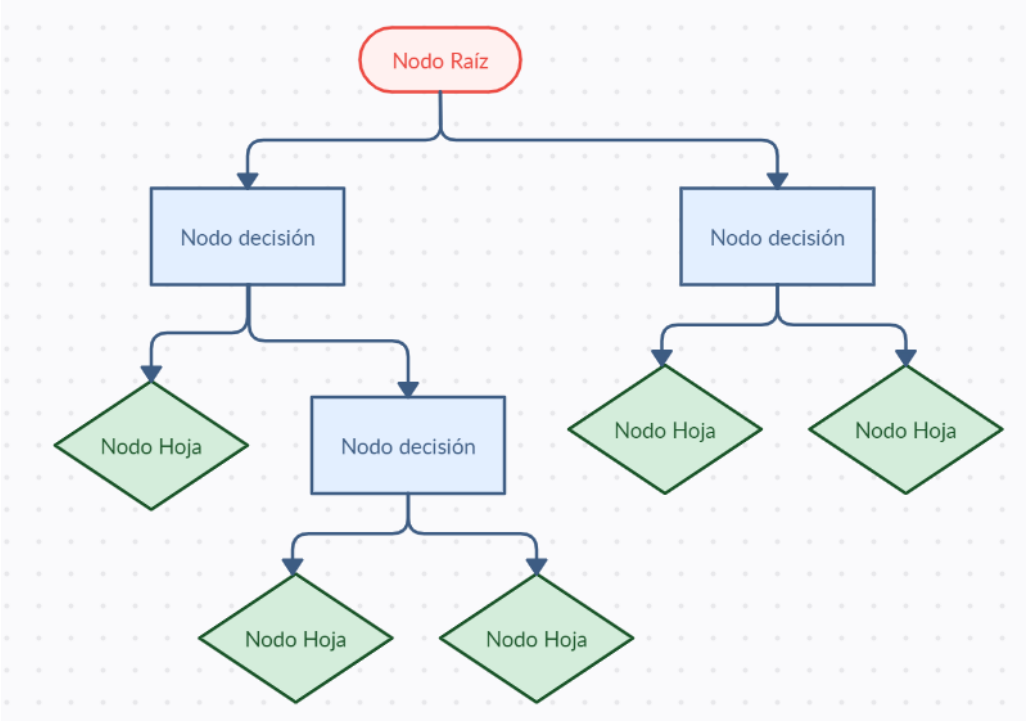


Figura 17. Árbol decisión 5 clases.

3.2.2.4.1. Función de impureza

Cada nodo toma decisiones con el fin de clasificar el objeto de entrada. Dichas decisiones, son tomadas según el criterio de minimización de la impureza, evaluando la calidad de la elección tomada. Una impureza de 0 significa orden global frente a una impureza de 1 que significa desorden. Algunas funciones de impureza son el índice de Gini, que mide la impureza de un nodo y la entropía, que mide el desorden de un sistema [56].

3.2.2.4.2. Hiperparámetros

Es común llegar a un estado de sobreentrenamiento al implementar este algoritmo, ya que por defecto se crea un árbol de decisión tan complejo como sea necesario para adecuarse a los datos de entrenamiento. Para evitarlo, se pueden modificar distintos hiperparámetros:

- *Max_depth*: profundidad del árbol.
- *Min_samples_split*: número mínimo de muestras requerido para dividir un nodo.
- *Min_samples_leaf*: número mínimo de muestras por nodo hoja.
- *Max_leaf_nodes*: número máximo de nodos hoja (número de clases a categorizar).
- *Min_impurity_split*: mínimo de impureza para dividir un nodo.
- *Criterion*: función de impureza que se va a utilizar.

3.2.2.5. Máquinas de vector soporte (SVM)

Las SVM son capaces de encontrar la forma óptima de separar varias clases a través de la búsqueda del hiperplano que maximiza del margen de separación entre clases y que equidista del ejemplo más cercano de cada clase [57].

Los vectores soporte son los que definen la máxima separación del punto más cercano de cada clase al hiperplano separador. Vienen representados por la unión perpendicular al hiperplano entre este y el ejemplo más cercano.

En la Figura 18, el hiperplano correcto sería la recta roja, y los vectores soporte las amarillas.

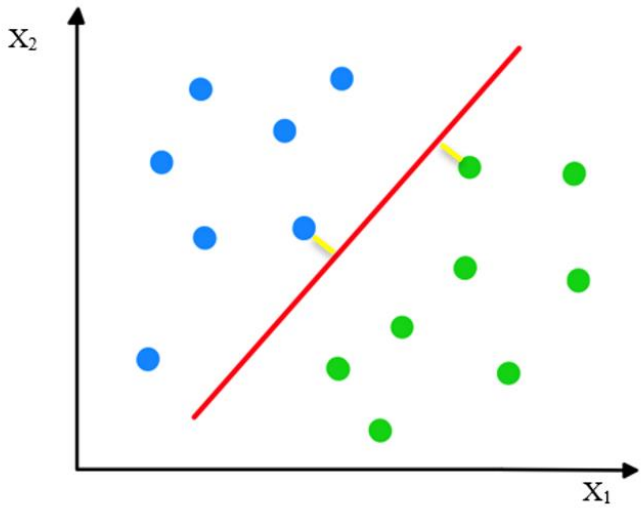


Figura 18. Ejemplo separación de clases mediante SVM

3.2.2.5.1. Aumento de la dimensión, kernels

El truco del *Kernel* es utilizado cuando las clases no se pueden separar por un hiperplano, es decir, no son linealmente separables. El truco reside en la invención de una nueva dimensión en la que sí se pueda encontrar dicho hiperplano.

En la Figura 19 se puede observar un ejemplo de esto. En la imagen de la izquierda resulta imposible separar las características verdes de las rojas a través de una recta. En cambio, si se añade una tercera dimensión, como en la imagen de la derecha, resulta sencillo dibujar un plano separador.

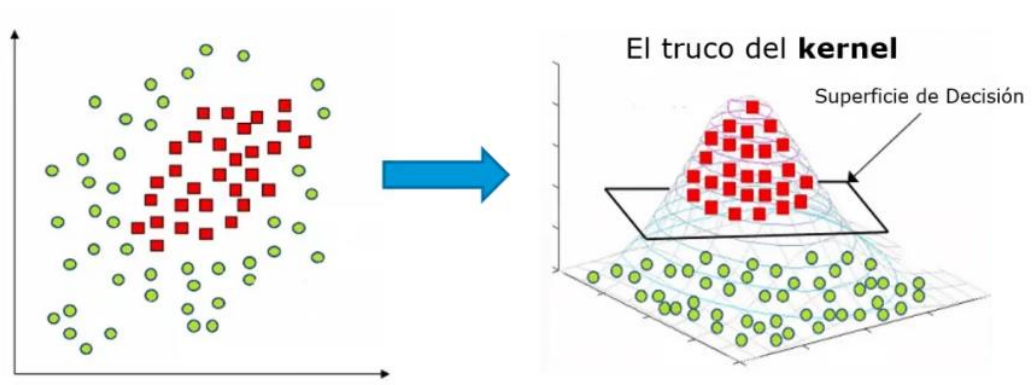


Figura 19. Truco del *kernel* en SVM. Disponible en [57].

3.2.2.5.2. Separación multiclases

Para casos distintos del binario, como es el del presente estudio en el que se pretenden separar 13 clases, se necesitarían más hiperplanos. Esto no es fácilmente generalizable por lo que se han desarrollado diferentes estrategias para la implementación de este algoritmo [58]:

One-versus-one: compara todas las posibles N clases dos a dos, generando un total de N(N-1)/2 de SVMs, asignando la entrada a la clase que ha sido asignada con más frecuencia.

One-versus-all: se compara cada clase con las restantes y la que resulte positiva es la clase a la que pertenece la entrada. Se necesitan un total de N SVMs.

DAGSVM: se inspira en el método *one-versus-one* tratando de reducir el tiempo de ejecución eliminando comparaciones innecesarias. Por ejemplo, en el caso concreto del clasificador de padel, si se compara un golpe de derecha con un golpe de revés y se obtiene como resultado que la entrada es un golpe de derecha, todas las operaciones restantes, en las que se vaya a comparar con un golpe de revés, resultarán eliminadas.

3.2.2.5.3. Regularización

Es controlada por el hiperparámetro C y consiste en indicarle al algoritmo la preferencia de generalizar bien para la mayoría de los casos, ya que es bastante común que algunos datos no estén etiquetados correctamente o que tengan ruido. Se calcula de la siguiente manera [57]:

$$C = \frac{1}{\propto} \tag{12}$$

siendo $\propto$ el hiperparámetro que indica la relación simplicidad frente a rendimiento del modelo [59].

3.2.2.6. K vecinos más próximos (KNN)

La idea del algoritmo de los k vecinos más próximos se basa en asignar el objeto de entrada a la clase que posea una mayor cantidad de vecinos más parecidos a él dentro del conjunto de entrenamiento [60].

Consiste en dividir el espacio en regiones conocidas con los datos etiquetados del entrenamiento. La distancia entre los vecinos etiquetados y la entrada que se quiera clasificar se mide, comúnmente, con la distancia euclídea: en un espacio euclídeo n-dimensional, la distancia euclídea entre dos puntos A = (a_1,a_2…a_n) y B = (b_1,b_2…b_n) viene dada por [61]:

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

siendo:

- d = distancia euclídea
- A = (a_1,a_2…a_n)
- B = (b_1,b_2…b_n)
- n = nº de dimensiones del espacio euclídeo

(13)

En función de los k vecinos que se deseen escoger, puede resultar una clasificación u otra. Por ejemplo, en la Figura 20, se puede observar que si se coge una vecindad de 3 (k = 3) se clasificaría el punto verde como círculo rojo. En cambio, si se escoge una k de 5, la clasificación sería como cuadrado azul.

Se podría decir que el hiperparámetro k determinará la clase del objeto por lo que es de gran importancia obtener su valor óptimo.

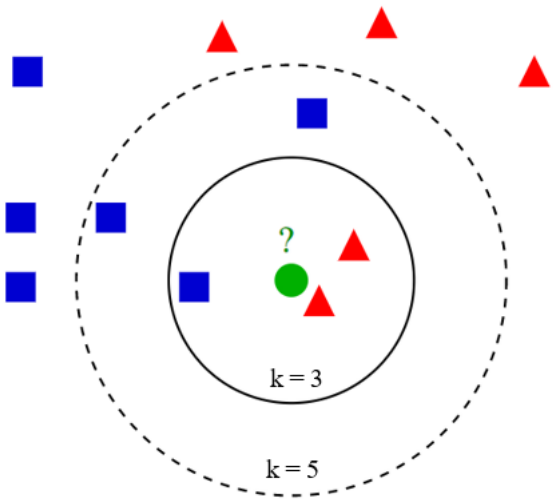


Figura 20. Algoritmo KNN. Disponible en [60].

Tras la explicación de esta idea básica, no resulta complicado la introducción de variantes que mejoren la clasificación. Una opción es la ponderación de la contribución de cada vecino en función de la distancia entre él y el objeto de entrada que se quiera clasificar, dando mayor peso a los vecinos más cercanos frente a los que se encuentran a una mayor distancia. Una forma de ponderar el peso de cada vecino es en función del cuadrado

inverso a sus distancias. Siguiendo con los datos expuestos en el ejemplo de la distancia euclídea, el peso del vecino B con respecto al punto A vendría dado por [60]:

$$w_b = \frac{1}{d(A,B)^2} \quad (14)$$

siendo w_b el peso y d(A,B) la distancia euclídea entre A y B.

Por lo tanto, la clase asignada sería la que maximice el valor de la suma de los pesos de sus representantes.

En el ejemplo de la Figura 20, al escogerse k = 5, los triángulos rojos obtendrían unos pesos mayores que los de los cuadrados azules. Esto implicaría que el círculo verde sería clasificado como triángulo rojo y no como cuadrado azul.

4. SISTEMA DE RECOPILACIÓN DE DATOS

4.1. Elementos hardware y software del sistema

A lo largo de este apartado, se expondrá el sistema utilizado para la obtención de datos, aunque para ello, se debe de introducir el contexto del juego añadiendo, brevemente, en qué consiste este deporte. Se juega por parejas en una pista como en la de la Figura 21, aunque existen también pistas individuales. En concreto, la pista de la Figura es el modelo WPT Challenger, pista oficial de la competición de padel más importante del mundo. El área de juego debe de tener unas dimensiones de 10x20 m y ambos lados de la pista deben de presentar simetría. Entre la pared y la línea de saque hay 3 m de distancia y las líneas del suelo deben de tener un ancho máximo de 5 cm. Se puede observar de una forma más clara las medidas en la Figura 22.

Los fondos pueden ser de pared, cristal o plástico con una altura de 3 m, el 4º m del fondo es de valla metálica, al igual que la mayor superficie de las "paredes" laterales. Los accesos a la pista se hacen por dos puertas laterales a la altura de la red, pudiéndose utilizar también durante los puntos [15].

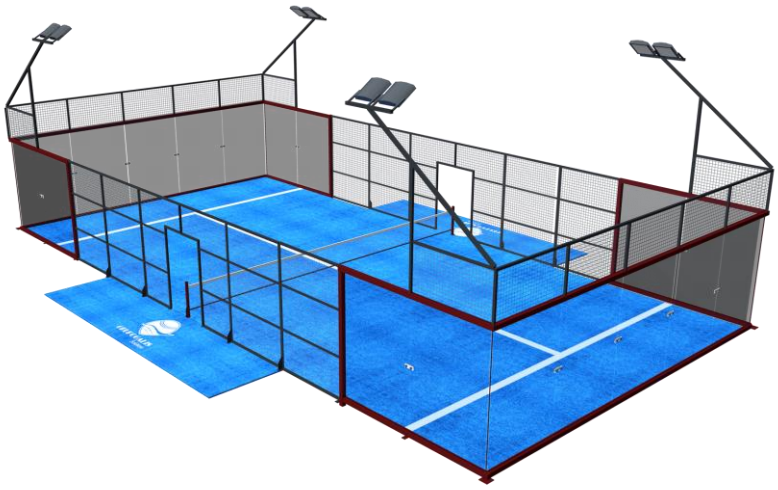


Figura 21. Pista de padel. Disponible en [16].

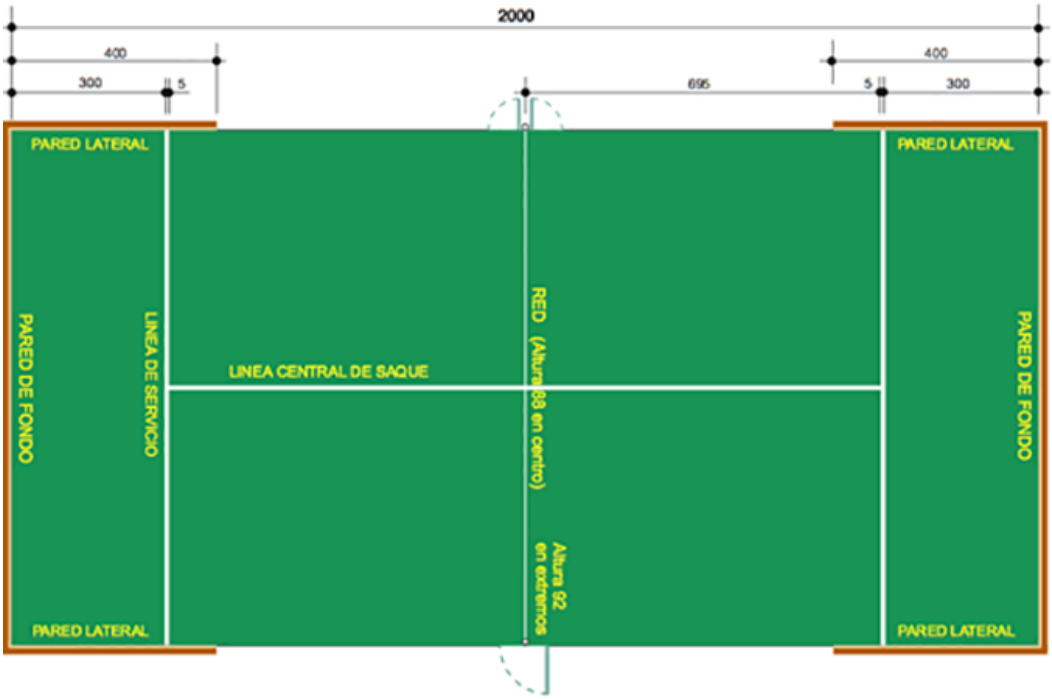


Figura 22. Dimensiones pista de pádel. Disponible en [17].

El objetivo es hacer que la pelota bote en el campo del oponente sin que lo haya hecho anteriormente en tu campo más de una vez. El sistema de puntuación es muy similar al utilizado en el tenis, con pequeñas variaciones como el punto de oro, aunque no es lo que nos atañe actualmente. Además, se juega con palas y pelotas como las disponibles en la Figura 23. Según el reglamento oficial, las pelotas deben de ser de color amarillo, de forma esférica con un diámetro de entre 6.35 y 6.77 cm, un peso de 56 a 59.4 g y un rebote desde los 2.54 m entorno a los 135/145 cm [17]. Con respecto a las palas, tienen que tener 45.5 cm de largo, 26 cm de ancho y 3.8 cm de grosor de perfil y debe de incluir una correa de unos 35 cm con el fin de prevenir accidentes y evitar la realización de trampas, cambiando de mano durante el juego [17].

Figura 23. Palas y pelotas de pádel. Disponible en [18].

El problema de la clasificación de los golpes de padel se puede asemejar a una clasificación de movimientos de muñeca de la mano con la que el deportista agarra la pala. Para que los algoritmos de aprendizaje automático realicen dicha clasificación, necesitan de una base de datos representativa que recoja una considerable cantidad de golpes.

Para la creación de dicha base de datos, se necesita un dispositivo capaz de recolectar tanto las velocidades como aceleraciones de la mano del deportista. Para ello, se utilizarán un giroscopio y un acelerómetro que proporcionarán 6 GDL: tres velocidades angulares y tres aceleraciones lineales, respectivamente. En adición, un sistema de interpretación de la información recogida por los sensores es necesario, eliminando información innecesaria y guardando los diferentes golpes de forma estándar.

La explicación del sistema de recolección de datos, se dividirá, por un lado, en los dispositivos hardware y, por otro, los componentes software:

4.1.1. Hardware

4.1.1.1. Raspberry pi

Es un ordenador de formato y precio reducidos cuyos diseños se basan en el hardware libre. Es común utilizarlo para el desarrollo de pequeños prototipos. Dispone de su propio sistema operativo Raspberry Pi OS, aunque también es conocido como Raspbian [19].

El modelo utilizado para este proyecto ha sido Raspberry pi 4, con una memoria RAM disponible de 4 GB.

4.1.1.2. IMU

Es una plataforma de medición inercial, compuesta por un sistema electrónico que contiene varios sensores: acelerómetros y giroscopios, capaces de recibir información dinámica del objeto al que se le aplica el movimiento. Los acelerómetros, que medirán la aceleración lineal de los tres ejes y giroscopios, que cuantificarán la orientación de un objeto, es decir, las velocidades angulares de los tres ejes. Algunas pueden incorporar también un magnetómetro, que medirá la fuerza y dirección del campo magnético, aunque no se hará función de él en este trabajo [20].

Como IMU se utilizará el *Sense Hat*, disponible en la Figura 24, dispositivo conectado a la Raspberry pi a través del GPIO. La verdadera IMU va incorporada en este dispositivo y es la LSM9DS1.

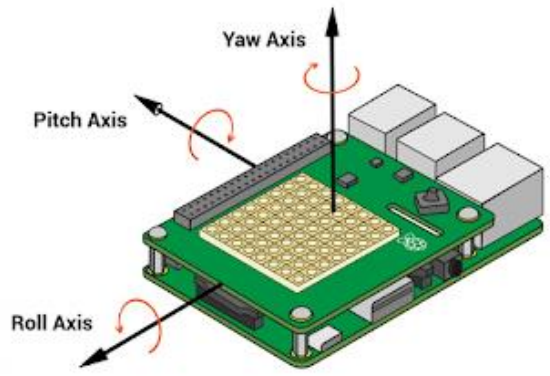


Figura 24. Sense Hat conectado a Raspberry pi con direcciones ejes. Disponible en [20].

4.1.1.3. Alimentación Raspberry pi

La Raspberry pi requiere de una fuente de alimentación de 5V 3ª mediante un puerto USB-C. Puesto que el jugador debe de moverse con libertad por la pista se ha elegido una batería portátil con las que se cargan los teléfonos móviles actuales.

4.1.1.4. Soporte

Se ha utilizado un soporte diseñado en 3D por el predecesor de este trabajo, Guillermo Cartes Domínguez, con unas dimensiones de 99x68x24 mm y un peso de 103 g. Se une a la Raspberry pi mediante tornillos y en la parte inferior se une a la muñeca con velcros. Además, para la toma de datos, se unirá a la muñeca colocando una muñequera sobre esta y cinta aislante para asegurar el cable de la alimentación. Este soporte no dificulta los

movimientos a la hora de realizar golpes de padel ya que, la mayoría, no requieren de apenas giros de esta articulación.

Figura 25. Soporte + *Sense Hat* + Raspberry pi + Cable alimentación. Disponible en [7].

4.1.2. Software

4.1.2.1. Python

Python es el lenguaje de programación utilizado. Se ha elegido puesto que presenta un lenguaje flexible, orientado a objetos y de alto nivel. Además, este utiliza paquetes y módulos lo que fomenta la reutilización y modularidad de código [21]. Su abundancia de librerías de las que dispone, hace que sea adecuado para este proyecto. Por lo que se ha escogido tanto para el sistema de toma de muestras como para los algoritmos de clasificación.

4.1.2.2. VNC Viewer

Puesto que la Raspberry pi es un ordenador independiente al cual se necesita acceder, se hará a través de *VNC Viewer*. Un programa que, a través de un cliente multiplataforma, permite tomar el control remoto del ordenador servidor, en este caso la Raspberry pi [22].

Para llevar a cabo dicha conexión se puede hacer de dos formas: a través de un cable ethernet que conecta la Raspberry pi con el ordenador con el que se quiera controlar, o vía Wifi compartida por un teléfono móvil al que se conectarán tanto la Raspberry pi como el ordenador local. Para evitar el estorbo al deportista durante la realización de las pruebas, se conectarán via Wifi.

4.1.3. Sistema completo

A modo de unificación de las diferentes partes el dispositivo completo de toma de datos funciona de la siguiente manera: El soporte de toma de datos con la Raspberry pi y el *Sense Hat* con la IMU incorporada se colocan en la muñeca del deportista sujetándolos con una tira de velcro. Además, se pondrá una muñequera encima para aportar seguridad y evitar movimientos innecesarios durante los golpeos. La Raspberry pi se alimentará a través de un cable que irá por dentro de la camiseta del deportista, rehuyendo de posibles enredos, y se conectará a la batería portátil alojada en su bolsillo.

Los datos recogidos por la IMU se almacenarán en la Raspberry pi, accediendo a ellos a través de *VNC Viewer* y un ordenador. Todos estos conectados al mismo Wifi, proporcionado por un teléfono móvil. Se muestra de una forma más visual en la Figura 26.

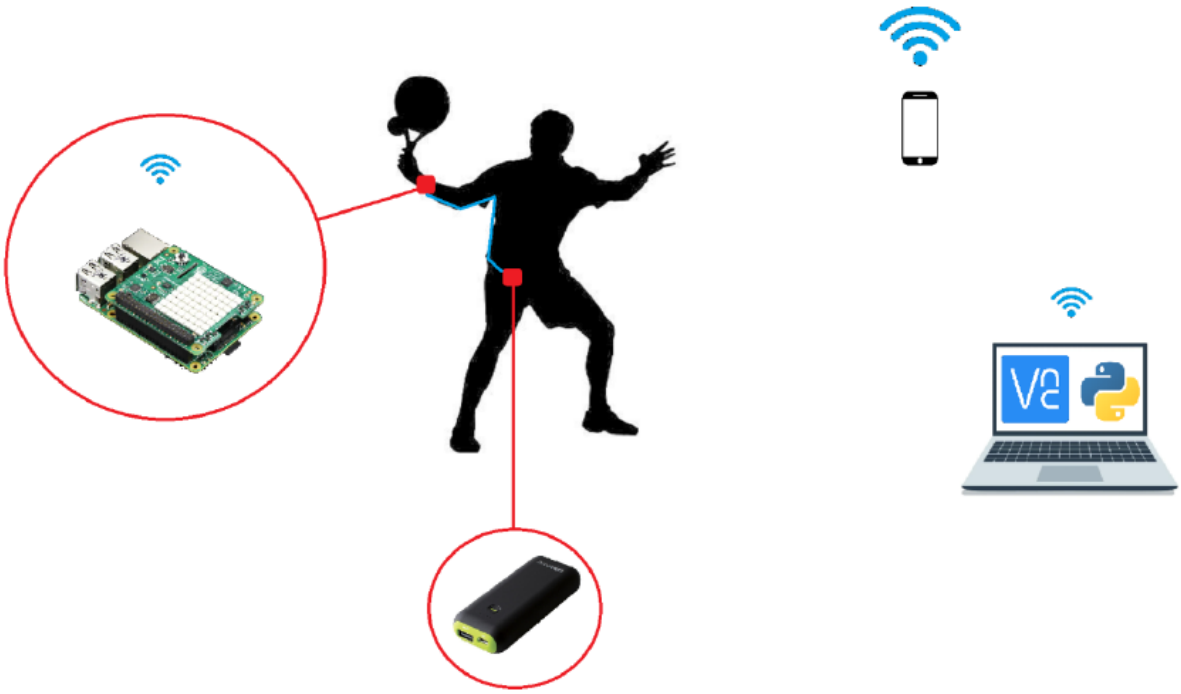


Figura 26. Sistema completo de recolección de datos. Disponible en [7].

4.2. Generación del conjunto de datos o dataset

Se necesitarán tres personas para poder realizar el proceso de una forma rápida y ágil:

- *Deportista*: será el encargado de realizar los golpeos de la pelota durante la prueba. Llevará la Raspberry pi en la muñeca tal y como se ha explicado en la sección anterior. El requisito para realizar la prueba es conocer el deporte y saber ejecutar los golpeos.
- *Persona lanzadora de pelotas*: pondrá la pelota en juego, lanzándola convenientemente al deportista según la prueba que se esté grabando. Deberá de conocer el deporte puesto que la colocación de la pelota en el punto exacto es complicada. Material técnico como un carro de pelotas facilita el proceso.
- *Responsable del registro de datos*: supervisará la prueba y comprobará que la toma de datos transcurre de forma normal. También será el encargado de establecer las conexiones y de darle al botón de ejecución del programa al inicio de la prueba. Además, grabará el proceso en vídeo colocando el teléfono, desde el que se compartirá el Wifi, en un trípode, quedando constancia visual.

Figura 27. Proceso de toma de muestras con las diferentes personas implicadas

Este proceso también se puede llevar a cabo con dos personas ya que, una sola se puede encargar tanto de lanzar las pelotas como de vigilar que el programa esté funcionando correctamente, aunque resulta más tedioso y lento.

4.2.1. Proceso de toma de datos

La prueba se dividirá en tantas partes como golpes se vayan a estudiar. En este trabajo serán un total de 13 y serán explicados en la siguiente sección. Durante cada prueba el jugador solo podrá hacer el tipo de golpe que se está estudiando y los resultados se guardarán en un .csv con el nombre del golpe realizado. Por ejemplo, si se está trabajando el golpeo de revés, el jugador durante la duración de la prueba (minuto y medio aproximadamente) tan solo podrá darle a la pelota de revés y los resultados se guardarán en "reves.csv". El periodo de muestreo será de 20 Hz.

4.2.2. Tipos de golpes

Como se ha mencionado en el apartado anterior, la cantidad de golpes a estudiar serán 13. Cabe destacar que en estos movimientos no se ha considerado el efecto de la pelota. Sí que se hará diferencia en los momentos de golpeo de la pelota, es decir, si esta ha tocado antes la pared o no. Los distintos golpeos con su número de referencia en la base de datos son los siguientes:

0 → *Fondo derecha*

Se golpea la pelota por el lado que el jugador sostiene la pala después de que la pelota haya tocado primero el suelo. Se suele realizar por detrás de la línea de servicio.

1 → *Fondo revés*

Es el movimiento complementario al anterior, ya que se golpea por el lado contrario al brazo dominante del jugador, por tanto, se debe utilizar la cara contraria de la pala. Al igual que el anterior, se ejecuta una vez la pelota haya botado y suele ser tras la línea de servicio.

2 → *Derecha con pared*

Es un golpe defensivo y se ejecuta una vez haya tocado la pared de fondo por el lado del brazo con el que el jugador sostiene la pala.

3 → *Revés con pared*

Al igual que el anterior, es un golpe defensivo que se realiza cuando la pelota ha tocado la pared de fondo y hacia adelante. La diferencia es que se realiza por el lado contrario al brazo hábil del deportista.

4 → *Globo de derecha*

El propósito de este golpeo es arrebatar la red a los rivales lanzando una pelota alta por encima de sus cabezas, por lo que es un movimiento que se realiza desde abajo hacia arriba por el lado dominante del jugador.

5 → *Globo de revés*

Tiene el mismo objetivo que el globo de derecha y se realiza de la misma forma, aunque este se ejecute por el lado opuesto, es decir, por el lado contrario al que el jugador sostiene la pala.

6 → *Globo de derecha con pared*

Con este golpe se busca mandar a los rivales al fondo de la pista por detrás de la línea de servicio. Se realiza de la misma forma que el globo de derecha con la diferencia que en este caso se golpea la pelota una vez haya rebotado en la pared.

7 → *Globo de revés con pared*

El objetivo es el mismo que el del golpe anterior, arrebatar la red a los rivales y se ejecuta de la misma manera aunque por el lado contrario al brazo dominante del jugador.

8 → *Volea de derecha*

El jugador que realiza el golpeo se coloca en la red y golpea la pelota por el mismo lado de su brazo hábil antes de que bote. Desde aquí se busca ganar el punto puesto que se está atacando.

9 → *Volea de revés*

El jugador que realiza el golpeo se coloca en la red y golpea la pelota por el lado contrario de su brazo hábil antes de que bote. Desde esta posición se busca ganar el punto puesto que se está atacando.

10 → *Bandeja*

El objetivo de este golpeo es no perder la red, puesto que es en esta zona donde se ganan la mayoría de puntos. Se realiza cuando se recibe una pelota alta más bien a media pista y está entre la volea y el remate. Se realiza por el lado dominante del jugador.

11 → *Remate*

La intención principal de este golpe es terminar el punto. Se realiza cuando se recibe una pelota alta no muy lejos de la red y se golpea por encima de la cabeza con cierta velocidad, poniendo todo el peso del cuerpo sobre el golpe.

12 → *Saque*

Se utiliza para poner la pelota en juego y el propio deportista se coloca la bola. Se realiza detrás de la línea de servicio desde un lado de la pista y se debe de lanzar en cruzado sin tocar la verja. Por normativa no se debe de golpear por encima de la cintura.

Nota: se ha respetado el orden y el número de los golpes procedentes de la base de datos previa disponible en [7].

A pesar de que solo se han utilizado los golpes más comunes durante un partido existen otros como la víbora, el remate x3, el remate x4, las jugadas con las paredes laterales, las contraparedes, etc.

4.2.3. Características de los deportistas

Las características de los deportistas a destacar y que se han utilizado en la base de datos, por orden, han sido:

ID → número identificativo del deportista, es único para cada uno.
Sexo → masculino o femenino

Nivel → se ha hecho una clasificación en 5 niveles:

1. *Iniciación*: el jugador lleva menos de un año jugando.
2. *Amateur*: el jugador lleva más de un año jugando regularmente y no está federado.
3. *Intermedio*: el jugador está federado y es considerado, por entrenadores, nivel medio.
4. *Avanzado*: el jugador está federado y es considerado, por entrenadores, nivel alto.
5. *Profesional*: el jugador es internacional, deportista en el WPT, competición más importante de padel

del mundo.

Mano → diestro o zurdo

Revés → puede tomar valores de 1 o de 2 en función de cómo realice el revés el deportista. Lo más común es el reves con una mano, aunque hay jugadores, sobre todo *beginners* procedentes del tenis que lo realizan con ambas manos. La realización con dos manos limita el movimiento por lo que se espera apreciar una distinción en los datos.

Altura (cm) → altura del deportista
Edad (años) → años del deportista
Número de golpes → cantidad de golpes detectados de cada deportista

4.2.4. Conjunto de datos

Cabe destacar que se disponía de una base de datos con un total de 12 deportistas y 2328 golpes realizados por jugadores exclusivamente diestros [7]. Se ha ampliado introduciendo jugadores tanto diestros como zurdos con las siguientes características:

Tabla 1. Características de los deportistas añadidos a la base de datos

ID deportista	Sexo	Nivel	Mano	Revés	Altura (cm)	Edad (años)	Número Golpes
13	1	1	0	1	158	53	164
14	0	2	0	2	181	18	167
15	1	1	1	1	170	26	157
16	1	4	0	1	159	21	159
17	1	3	0	1	168	50	126
18	0	4	1	1	180	58	138
19	1	2	0	1	157	22	156
20	0	1	1	1	170	23	118
21	0	3	1	1	185	21	176
22	0	4	1	1	177	23	169
23	1	3	0	1	165	24	168

Durante la realización de las pruebas no se han realizado la misma cantidad de golpes ya que, por ejemplo, los golpes que requieren el uso de la pared necesitan más tiempo para ejecutarlos que las voleas.

Añadiendo a la base de datos los movimientos de los nuevos jugadores se tienen un total de 4002 golpes. La cantidad de golpeos totales, tanto antiguos como nuevos, por tipo de golpe almacenados en la base de datos son los mostrados en la Figura 28.

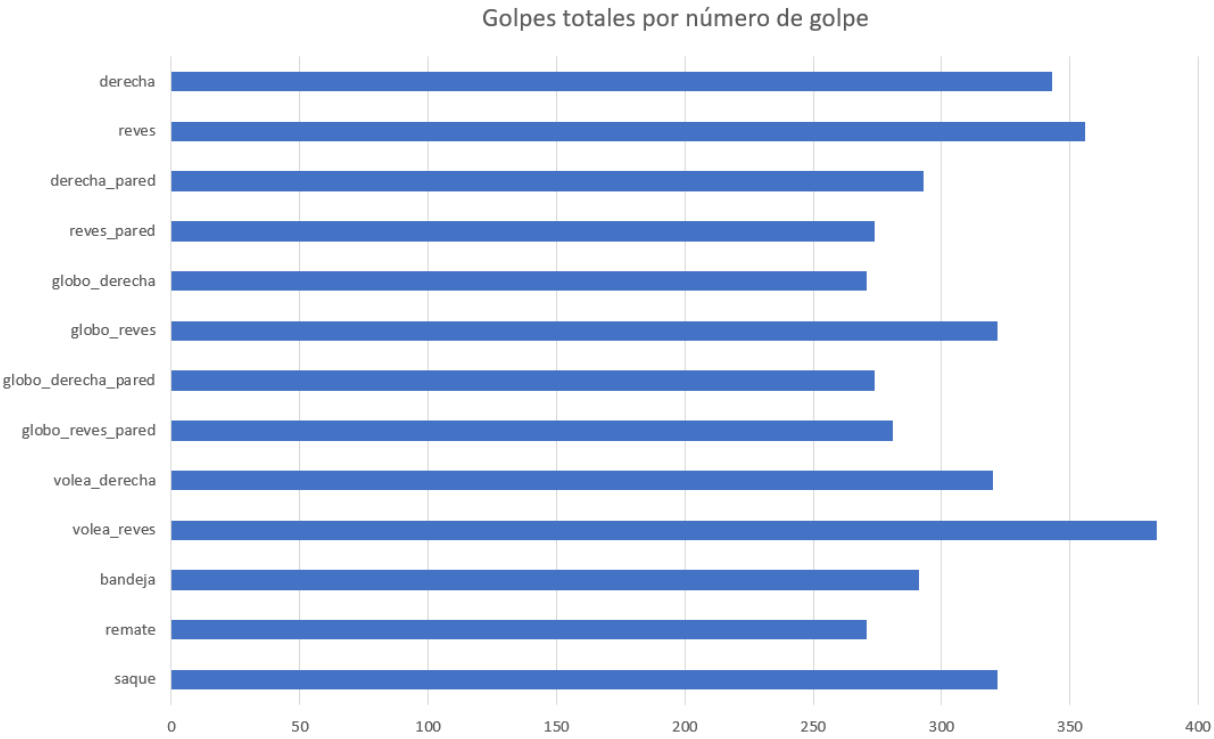


Figura 28. Golpes totales en la base de datos por tipo de golpe

4.2.5. Identificación de golpes

El acelerómetro y el giroscopio están continuamente recogiendo datos desde que empieza la prueba hasta que finaliza con un periodo de 20 Hz, por lo que no se hace distinción entre los datos que provienen del golpe, el tiempo de espera hasta la recepción de la pelota, etc.

Una pregunta interesante antes de tomar decisiones sobre los datos es: ¿cuánto dura un golpe? En este proyecto se ha considerado que el golpe inicia con la preparación del golpe y finaliza con el movimiento posterior al impacto de la pelota. Falta identificar en qué momentos del .csv se producen.

Cuando se prepara un golpeo los 6 GDL sufren cambios significativos por lo que si estos superan cierto umbral significa que se va a golpear la pelota y se tendrán que guardar los datos. Es por este motivo, por el que es razonable que la condición de detección de preparación del golpeo se da cuando simultáneamente uno de los 3 GDL del giroscopio y acelerómetro superan su umbral.

Estos umbrales se pueden superar más de una vez durante un mismo golpeo y se detectarán más de un golpe erróneamente puesto que tan solo es un movimiento. Por tanto, se establece una variable de desactivación de búsqueda de un golpe mientras no se haya guardado el último golpe detectado.

Los valores estimados para cada umbral son:

$$Ua = 3\,Gs$$
$$Uv = 5\,rad/s$$

Siendo U_a el umbral de aceleración y U_v el de la velocidad.

Una vez identificado el inicio del movimiento y desactivada la variable “búsqueda de golpe nuevo”, surge la siguiente cuestión: ¿cuánto tiempo dura el golpeo y, por tanto, hay que volver a activar dicha variable? Fijar la duración del golpeo es un asunto no objetivo puesto que depende directamente de la naturaleza del golpe. Una volea es un movimiento más rápido que un remate, el cuál requiere de una mayor preparación. La duración establecida para todos los golpeos es de 2 segundos.

Un ejemplo del detector de golpes se puede observar en la Figura 29, donde las líneas verdes indican la detección de un golpeo nuevo.

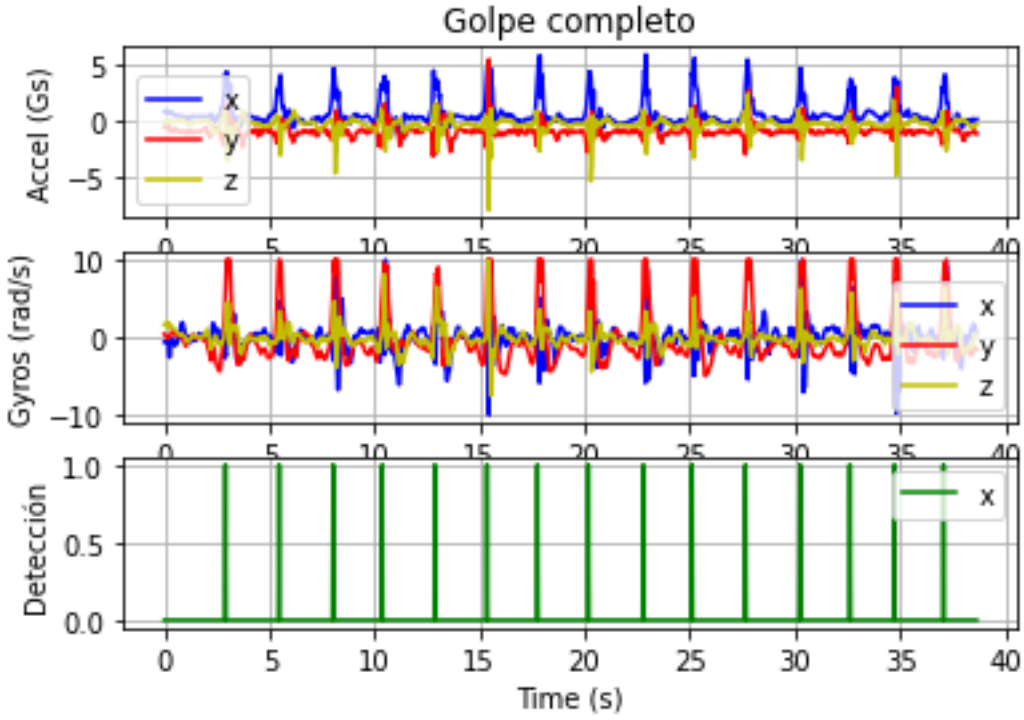


Figura 29. Ejemplo detector de golpes

Para comparar mejor los resultados del aislamiento de los golpes, se mostrarán algunos ejemplos. En la Figura 30 se muestra un remate y en la Figura 31 un globo de revés.

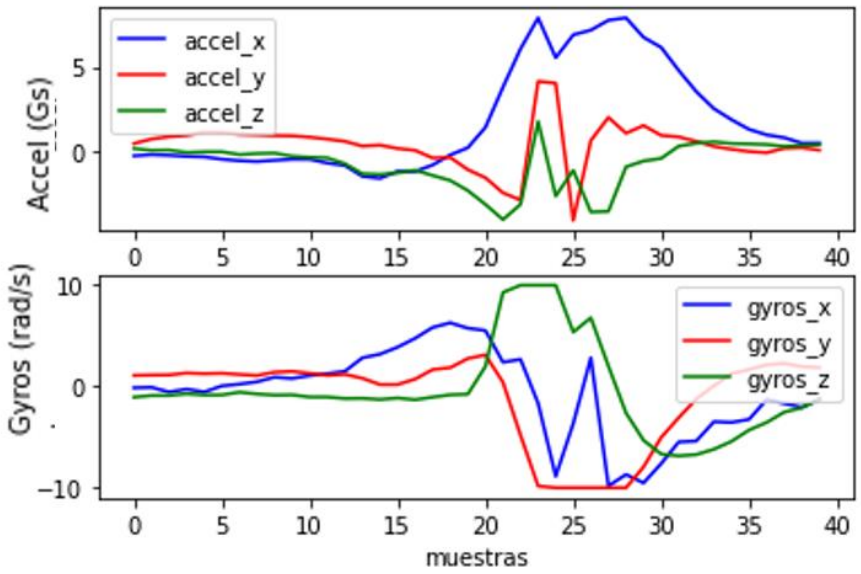


Figura 30. Datos del acelerómetro y del giroscopio en un Remate

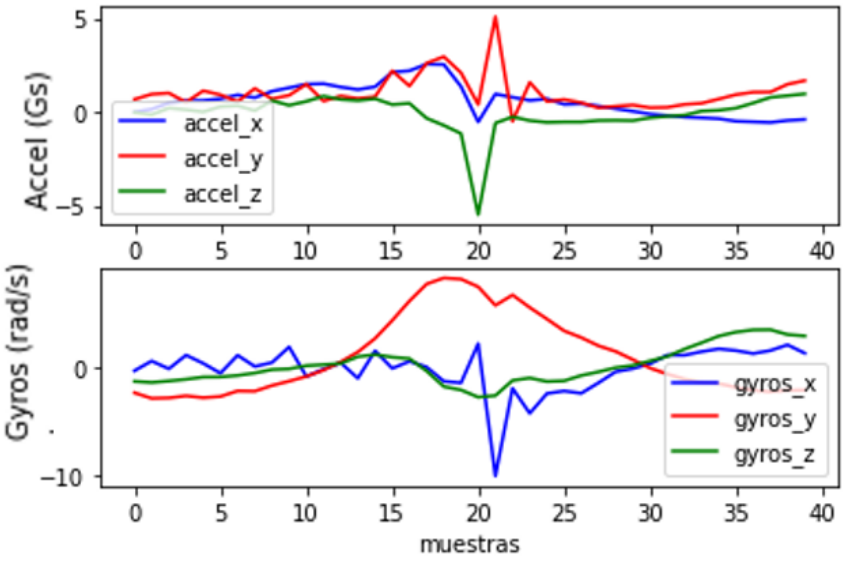


Figura 31. Datos del acelerómetro y del giroscopio en un Globo de Revés

Para observar la diferencia entre dos tipos de golpes, en la Figura 32 se han dibujado por separado las distintas aceleraciones y velocidades para un globo de revés, representado en color azul, y para un globo de derecha, en color rojo.

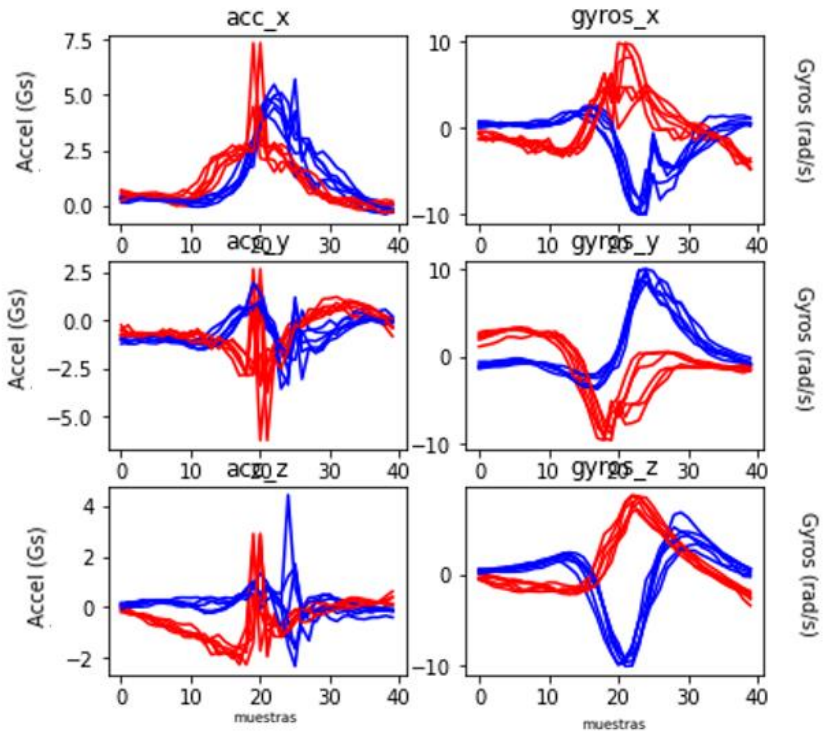


Figura 32. Comparación golpes: globo de derecha (rojo) vs globo de revés (azul)

Además, dependiendo del tipo de golpe se demuestra que las características del jugador pueden llegar a influir en gran medida en los datos del acelerómetro y del giroscopio.

Por ejemplo, en la Figura 33 y la Figura 34 dos jugadores zurdos realizan varios golpes de saque y en la Figura 35 y la Figura 36 son dos jugadoras diestras las que los realizan. Se muestra una clara diferencia en las aceleraciones en el eje x, ya que para los zurdos son positivas y para las diestras negativas, y en el eje z que pasaría justo lo contrario: positivas para las diestras y negativas para los zurdos.

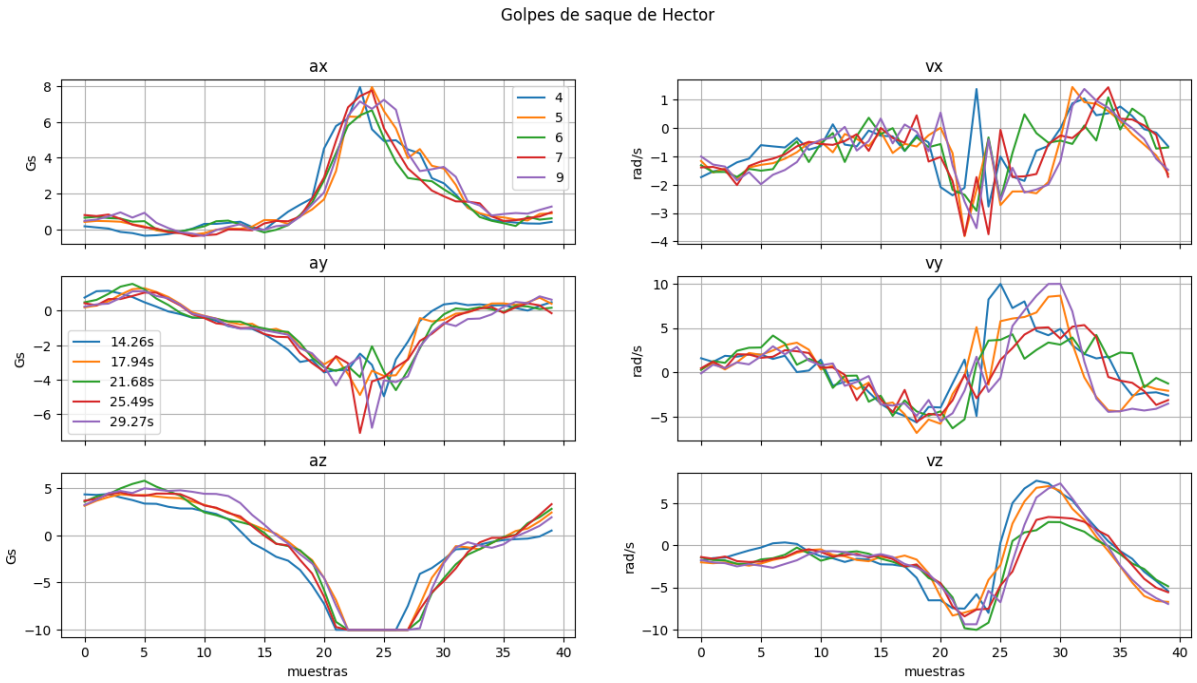


Figura 33. Golpes de saque realizado por jugador zurdo 1

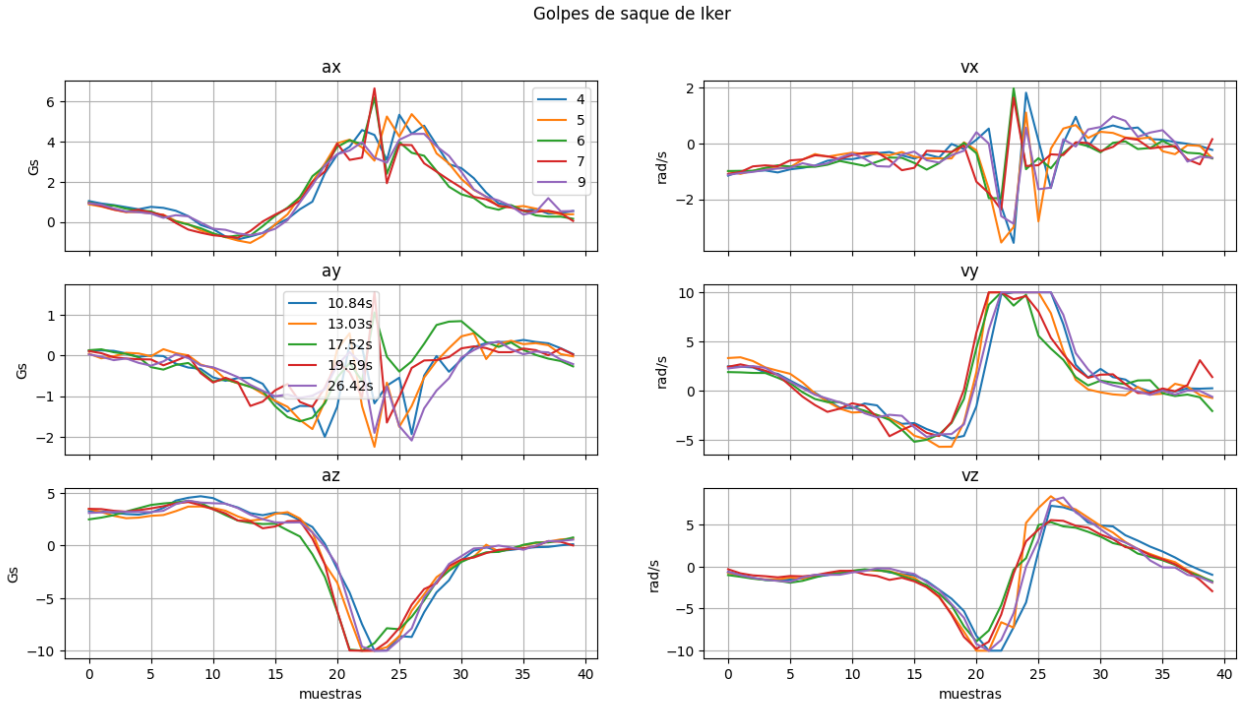


Figura 34. Golpe de saque realizado por jugador zurdo 2

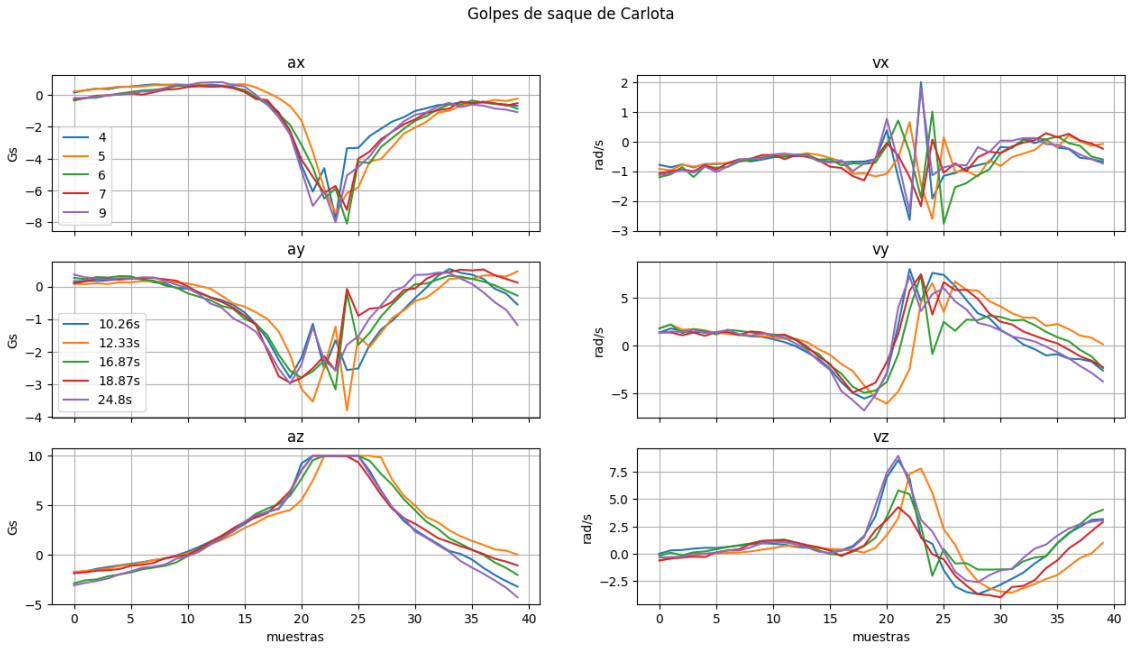


Figura 35. Golpe de saque realizado por jugadora diestra 1

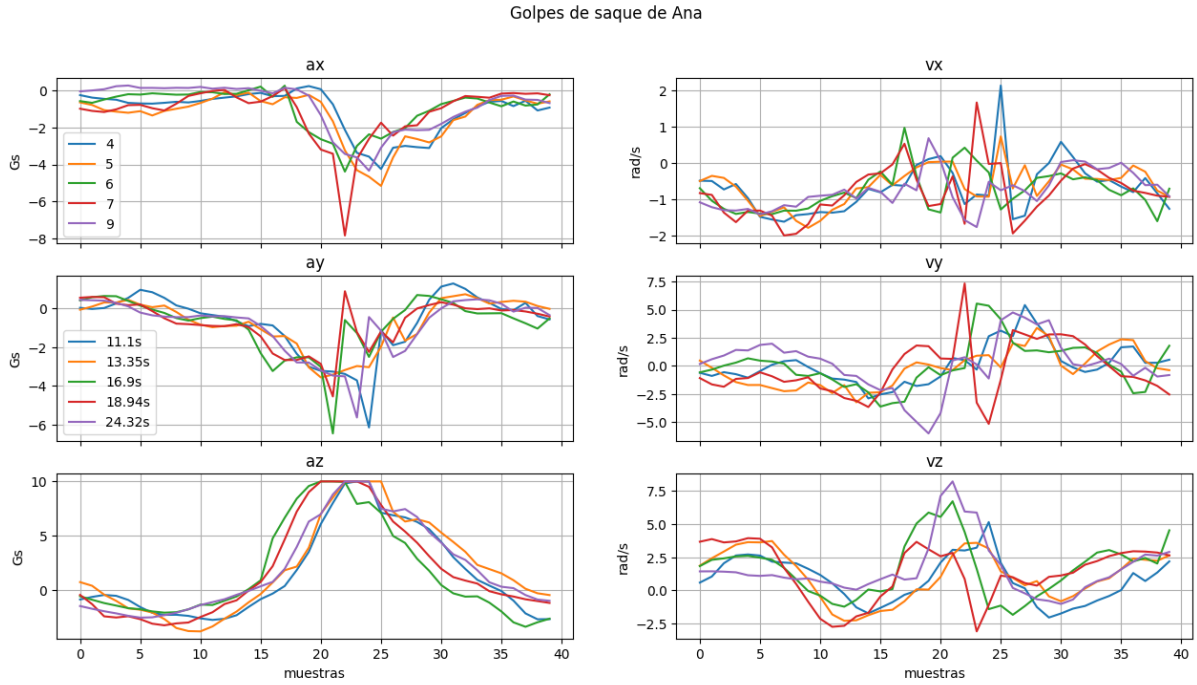


Figura 36. Golpe de saque realizado por jugadora diestra 2

Además, se comprueba, que a mayor nivel existe una mayor mecanización del golpeo por lo que, las aceleraciones y velocidades poseen unas curvas con respecto al tiempo de casi idéntico aspecto. Se puede observar en la Figura 35 los movimientos realizados por la jugadora Carlota, deportista profesional, presentan unas curvas muy similares frente a los realizados por Ana, en la Figura 36, que apenas lleva un año entrenando y cuyas curvas guardan una menor relación.

4.2.6. Creación base de datos temporal

En el estado actual del proyecto se tienen diferentes carpetas con nombres iguales a los de los deportistas que

han realizado las pruebas. Cada carpeta contiene los archivos .csv de los movimientos de cada jugador con el nombre del golpeo que representan. Cada .csv encierra distintos golpes de un mismo tipo de golpeo y cada uno se debe de componer, como se ha visto anteriormente, de 40 muestras (2 segundos a 20 Hz). Esto hace un total de 240 datos por golpe (6 GDL por 40 muestras). En la base de datos también deberán de guardarse las características del golpe como del deportista que lo ejecuta.

Para crear el dataset, tan solo habrá que recorrer todos los archivos de todas las carpetas de los distintos deportistas, guardando cada golpe con sus características correspondientes, identificadas con el nombre del archivo (tipo de golpe) y carpeta a la que pertenece (nombre del deportista). Para ampliar el dataset, será tan sencillo como añadir carpetas al directorio en el que se encuentren las demás.

El primer paso para la creación del dataset es recorrer los archivos del directorio indicado. Si es una carpeta, lee el directorio de la carpeta e identifica su nombre (el nombre del deportista) guardando sus características. Se recorren los archivos de esta nueva carpeta y si se encuentra con un .csv, identifica el nombre (tipo de golpe), detecta los golpes que haya en la prueba y guarda todos los datos.

El proceso finaliza cuando se han recorrido todos los archivos del directorio introducido inicialmente.

Para finalizar, en la Tabla 2 se muestra la forma que tendrá la base de datos. Compuesta por tantas filas como golpes totales haya y 250 columnas. Las 240 primeras son las muestras de cada GDL tanto del acelerómetro como del giroscopio, las 3 siguientes características del golpe y las 7 últimas características del jugador, explicadas en el apartado 4.2.3.

Dentro de las características del golpe se encuentra el *tipo de golpe* que será el número que define el tipo de golpe según el apartado 4.2.2, *número de golpe* que indicará el número de golpe realizado durante la prueba del deportista y el *tiempo de golpe* que representará el instante de tiempo en segundos en el que se ejecutó el movimiento durante la prueba.

Tabla 2. Base de datos temporal

Ax0…Ax39	**Ay0…Ay39**	**Az0…Az39**	**Vx0…Vx39**	**Vy0…Vy39**	**Vz0…Vz39**	**Tipo de golpe**	**Número de golpe**	**Tiempo de golpe**	**Sexo**	**Nivel**	**Mano**	**Revés**	**Altura**	**Edad**	**ID**
Golpe 1															
Golpe 2															
⋮															
Golpe N															

4.2.7. Creación base de datos en frecuencia

Puesto que una de las metas del presente trabajo es el entrenamiento de las redes neuronales en el dominio de la frecuencia, una vez obtenida la base de datos en el dominio del tiempo, se tendrá que transformar al dominio de la frecuencia.

Cada fila de la base de datos actual se compone de 6 señales temporales distintas, correspondiéndose cada una con cada GDL del acelerómetro y giroscopio. Como se vio en el capítulo 2, se debe de aplicar la FFT, a la señal temporal que se desee transformar al dominio de la frecuencia. Por tanto, para cada fila se tendrán que aplicar 6 FFT diferentes.

Como se vió en el capítulo 2, por el teorema de Nyquist, la frecuencia de muestreo debe de ser 2 veces mayor

que la frecuencia más alta de interés. Puesto que se ha muestreado las señales a 20 Hz, aplicando la FFT, tan solo se podrá reconstruir hasta una frecuencia de 10 Hz.

Además, para el entrenamiento de la red y, por tanto, para la construcción de la base de datos tan solo se tendrán en cuenta la magnitud en valor absoluto de las componentes obtenidas como resultado de la aplicación de la FFT.

El resultado de la aplicación de la FFT, según lo explicado en el capítulo 3, debe de ser una señal simétrica con una frecuencia máxima de 10 Hz. En la Figura 37, se muestra la aceleración en el eje x de un golpe de derecha tras la transformación y en la Figura 38 la aceleración en ese mismo eje de un globo de revés, observándose así la diferencia entre ambos.

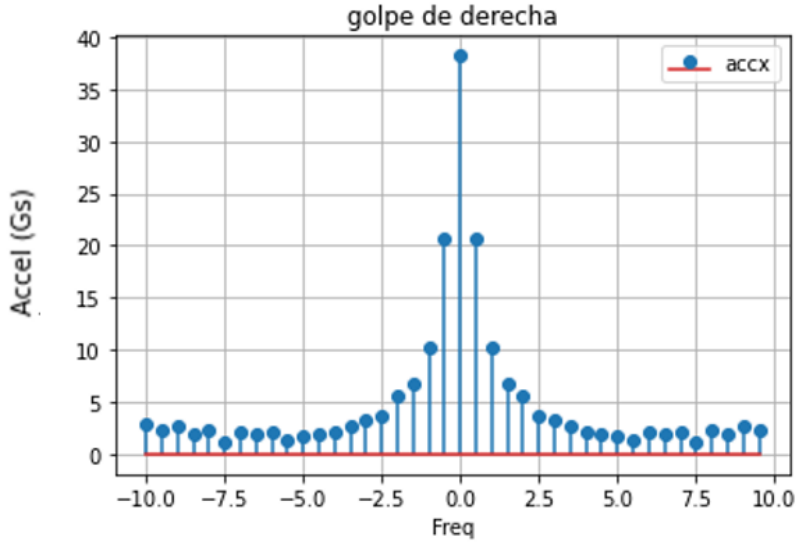


Figura 37. Aceleración eje x de un golpe de derecha en el dominio de la frecuencia

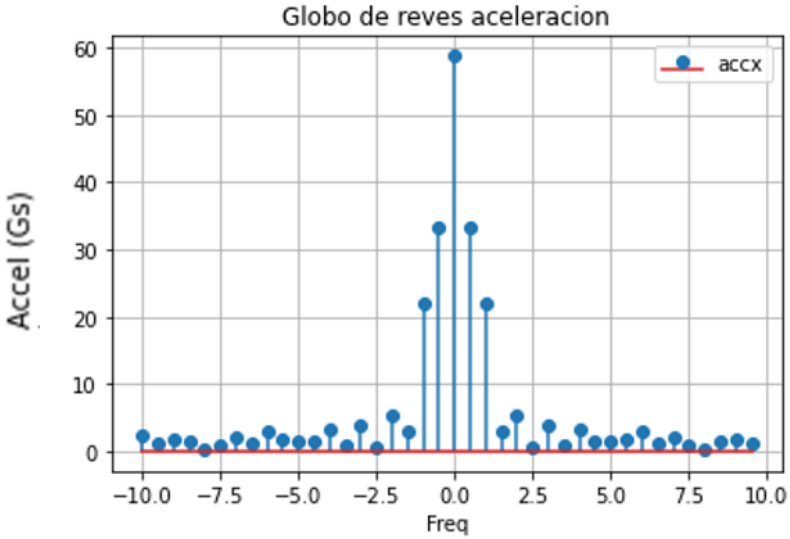


Figura 38. Aceleración en el eje x de un globo de revés en el dominio de la frecuencia

Debido a su simetría, no tiene sentido introducir como entrenamiento a la red neuronal todo el resultado de la FFT si no solo las frecuencias positivas, mostrado en la Figura 39.

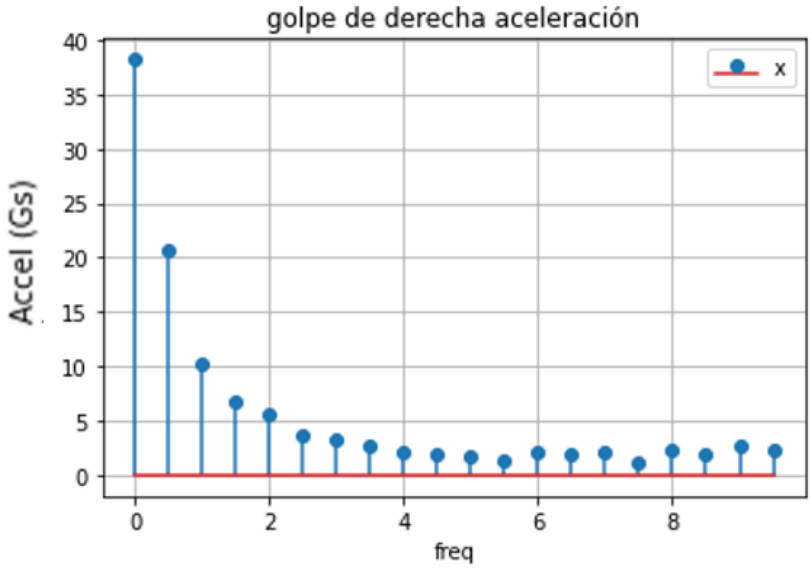


Figura 39. Datos efectivos en frecuencia aceleración eje x golpe de derecha

La base de datos en el dominio de la frecuencia posee una considerable cantidad inferior de columnas con respecto a la temporal ya que, por cada GDL de cada golpe se introducen la mitad de las muestras. El nuevo conjunto de datos posee un total de 130 columnas frente a las 250 antiguas. Su morfología se muestra en la Tabla 3.

Tabla 3. Base de datos en frecuencia

Ax0…Ax19	Ay0…Ay19	Az0…Az19	Vx0…Vx19	Vy0…Vy19	Vz0…Vz19	Tipo de golpe	Número de golpe	Tiempo de golpe	Sexo	Nivel	Mano	Revés	Altura	Edad	ID
Golpe 1															
Golpe 2															
⋮															
Golpe N															

4.2.8. Bases de datos a utilizar

Para llevar a cabo los objetivos del presente estudio, se necesitan 4 bases de datos diferentes, dos de ellas en el dominio temporal y otras dos en el frecuencial.

Dentro de cada dominio se ha de hacer una distinción, ya que se pretende observar la influencia de la adición de datos recopilados en jugadores zurdos al conjunto. Por lo tanto, en una de ellas habrá que prescindir de dichos golpes.

Para ello, se implementa un simple algoritmo en las bases de datos que se quieran modificar: se realiza un bucle que recorra todos los golpes del conjunto. En la casilla mano se comprueba si hay un 0 o un 1. En el caso de que exista un 1, significa que el jugador es zurdo, por lo que se procederá a la eliminación de dicha fila. En cambio, si hay un 0 se pasará a comprobar directamente la siguiente fila.

En un diagrama de flujo se vería de la forma mostrada en la Figura 40.

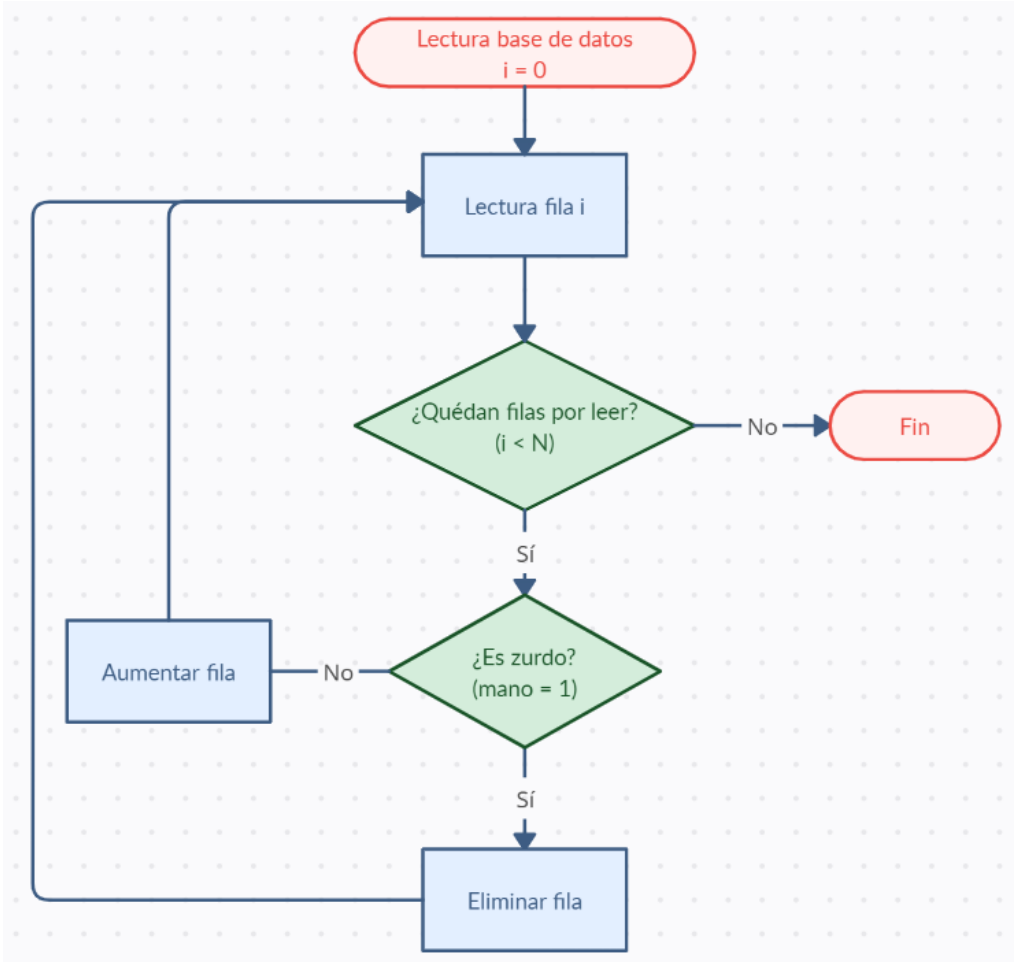


Figura 40. Diagrama de flujo para eliminar jugadores zurdos de la base de datos.

A modo resumen, se crearán 4 bases de datos distintas:

- Dominio temporal con jugadores diestros y zurdos.
- Dominio temporal con jugadores exclusivamente diestros.
- Dominio frecuencial con jugadores diestros y zurdos.
- Dominio frecuencial con jugadores exclusivamente diestros.

5. RESULTADOS DE COMPARACIÓN

No sabía cómo hacerlo, así que lo intenté, fallé, fallé, fallé y luego lo hice bien

-Elon Musk-

A lo largo de este capítulo se expondrán los resultados obtenidos con cada uno de los algoritmos implementados. Los códigos base utilizados han sido heredados del trabajo predecesor [7], aunque se han realizado modificaciones sobre estos para adaptarlos al dominio de la frecuencia, así como para la introducción y eliminación de jugadores zurdos y para el cálculo del tiempo de entrenamiento de los algoritmos y de clasificación de la muestra de entrada. Tanto los códigos como los *dataset* utilizados se encuentran en el repositorio *GitHub* disponible en [65].

Se utilizarán dos bases de datos diferentes, una en el dominio del tiempo y otra en el de la frecuencia ya que, entre los objetivos del presente trabajo, se encuentra la búsqueda del mejor algoritmo clasificador a través de la comparación del entrenamiento de distintos algoritmos, con los mismos datos etiquetados transformados.

Puesto que se ha ampliado la base de datos, se repetirán los experimentos realizados en [7], con el fin de mejorar los resultados.

La entrada que reciben los algoritmos, en este primer caso, son las series temporales de los golpes, es decir, por cada golpe se introducen 240 muestras (6 GDL por 40 muestras) además de la etiqueta del golpe y características varias del deportista que lo ejecuta, explicadas en la Tabla 1. En realidad, los únicos datos que necesita el algoritmo para ser entrenado son las 240 muestras del golpe y la etiqueta del mismo ya que, los golpes de padel se componen exclusivamente de movimientos de muñeca. La información adicional, podrá ser utilizada en trabajos futuros, explicados en detalle en el siguiente capítulo.

Los datos, se pueden introducir directamente en el algoritmo desde la base de datos o se les puede aplicar un preprocesado, *feature engineering*. El objetivo es extraer las características más notorias de los datos con el fin de facilitárselas como entrada al modelo. Las que se han escogido son el valor máximo, el medio y el mínimo de cada grado de libertad de la serie temporal del golpe, reduciendo el tamaño anterior de 240 muestras por golpe a tan solo 18. Esta técnica no se aplicará en el trabajo presente, debido a que, en el trabajo fin de máster antecesor [7], se demostró que daba peores resultados.

Otro aspecto que destacar es la división del conjunto de datos total ya que, como se vio en el apartado 3.2.1.2 de este mismo documento, no todos se usan para el entrenamiento del modelo. El 64% de los datos de la base serán destinados para el entrenamiento, el 16% para la validación y el 20% restantes para el testeo. En los casos donde no se realice la validación los datos de entrenamiento constituirán el 70% de los totales y los de testeo el 30%. La división de los datos es completamente aleatoria, aunque siempre será la misma, con la finalidad de que todos los modelos reciban la misma entrada.

5.1. Librerías utilizadas

El lenguaje utilizado para implementar los algoritmos es *Pyhton*. Se utilizan las siguientes librerías:

- *Tensor Flow – Keras*: tensor-flow es una librería de código abierto capaz de ejecutar de forma eficaz y eficiente gráficos de flujo, es decir, operaciones matemáticas representadas sobre nudos, cuyas entradas y salidas son vectores multidimensionales. Su principal aplicación está destinada para el aprendizaje automático y redes neuronales profundas. Keras es una biblioteca de redes neuronales escrita en Python, resultando ser la API de alto nivel de tensor-flow [62]. En este trabajo se utiliza la vesión 2.3.0 de tensor-flow y la 1.0.8 de keras.
- *Scikit-learn*: es una librería de libre acceso para *Python* que cuenta con algoritmos de clasificación. Una de las ventajas es la variedad de algoritmos y variedad de módulos, además es compatible con otras librerías como NumPy o matplotlib [63]. En este trabajo se utiliza la versión 0.24.2

Ambas librerías suponen una herramienta muy potente para la creación de los modelos matemáticos en este estudio. La plataforma en la que se lleva a cabo la programación es Spyder 4.2.5 en Anaconda y la versión de *Python* utilizada es la 3.8.

5.2. Exposición de resultados

Con la finalidad de medir la calidad del algoritmo se calculará el accuracy, es decir, el porcentaje de casos en los que el modelo ha acertado. Por tanto, cuanto más alto sea, mejor será el clasificador. Es la medida más directa de comprobar cómo de bien ha funcionado el algoritmo entrenado. Por ejemplo, en el clasificador binario de la Tabla 4, el accuracy se calcularía con la ecuación (15):

Tabla 4. Ejemplo de clasificador binario.

		Real	
		Positivos (1)	Negativos (0)
Predicción	Positivos (1)	TP	FP
	Negativos (0)	FN	TN

$$Accuracy\ (\%) = \frac{TP + TN}{TP + FP + TN + FN} * 100 \quad (15)$$

Siendo:

- TP, *True Positive*: la clase real del punto es positivo (1) y la predicha por el algoritmo también.
- TN, *True Negative*: la clase real del punto es negativo (0) y la predicha por el algoritmo también.
- FP, *False Positive*: la clase real del punto es negativo (0) y la predicha por el algoritmo es positivo (1).
- FN, *False Negative*: la clase real del punto es positivo (1) y la predicha por el algoritmo es negativo (0).

Resulta sencillo observar que el caso ideal sería obtener un accuracy del 100%, siendo 0 los campos FP y FN.

Para la presentación de resultados se mostrará la configuración que obtiene mejores resultados, así como un diagrama de bigotes para cada algoritmo. Además, se expondrá la matriz de confusión para la mejor accuracy, relacionando los valores reales de los datos (las etiquetas en este caso) con las predicciones realizadas por el clasificador. Se han establecido siglas para los nombres de los golpes, con el fin de mostrarlos en la matriz de confusión de una manera más clara y sencilla:

D → fondo Derecha
R → fondo Revés
DP → Derecha Pared
RP → Revés Pared
GD → Globo de Derecha
GR → Globo de Revés
GDP → Globo de Derecha con Pared
GRP → Globo de Revés con Pared
VD → Volea de Derecha
VR → Volea de Revés
B → Bandeja
RM → Remate
S → Saque

Tanto los diagramas de bigotes como las matrices de confusión que no se encuentren en este capítulo, están disponibles en los anexos.

La exposición de los resultados se dividirá en dos grandes bloques divididos a su vez en dos: por un lado, se entrenará a los algoritmos con la base de datos en el dominio del tiempo (al igual que en [7] pero con la ampliación de la base de datos) y por otro, el entrenamiento se realizará en el dominio de la frecuencia. Puesto que, anteriormente, no se habían introducido datos de jugadores zurdos, dentro de cada bloque se estudiarán los resultados para únicamente jugadores diestros y para ambos tipos de jugadores.

Cada modelo ha sido implementado un total de 10 veces. El accuracy mostrado es la media de los 10 obtenidos, aunque las matrices de confusión y los diagramas de bigotes son los de la configuración con mejores resultados. Además, los experimentos han sido realizados con un procesador Intel CORE i7.

Tras detallar la manera en que se mostrarán los resultados se procede a la exposición de los mismos.

5.3. Resultados en el dominio de la frecuencia

5.3.1. Jugadores zurdos y diestros

5.3.1.1. Redes neuronales densamente conectadas

En capítulos anteriores, concretamente en el 4.2.2.1. se expuso el funcionamiento este tipo de algoritmos. Los hiperparámetros a ajustar en el código son el batch size, el número de epochs y el número de filtros o neuronas por capa. Puesto que la combinación de configuraciones puede ser infinita, se realizará lo que se conoce como búsqueda de rejilla (*grid search*). La base de datos se dividirá asignando un 64% de estos para entrenamiento, el 16% para validación y finalmente el 20% restante para el testeo.

5.3.1.1.1. Dos capas densamente conectadas

Para comenzar los experimentos, se prueba un clasificador con dos capas densamente conectadas, la segunda de ellas, la de salida, dispondrá de 13 filtros, coincidiendo este número con los tipos de golpes a clasificar. Además, su función de activación debe de ser de tipo *softmax*, ya que debe de devolver la distribución de probabilidad sobre las clases de salida. Cabe destacar, que antes de la primera capa se añade una capa de aplanamiento, llamada *flatten*, ya que las capas densan necesitan un vector como entrada.

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
flatten_37 (Flatten)	(None, 120)	0
dense_108 (Dense)	(None, 2000)	242000
dense_109 (Dense)	(None, 13)	26013

Total params: 268,013

Trainable params: 268,013

Non-trainable params: 0

La primera capa del modelo está formada por una función de activación ReLU y un total de 2000 neuronas, mientras que la segunda la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **73.946%** (+/-0.565), como se resume en el diagrama de bigotes, disponible en el anexo A4.1.1. En el diagrama de bigotes se muestra visualmente la distribución de los datos numéricos: cuartiles, mediana, extremos.

Por último, en la matriz de confusión se pueden observar las clases donde más dificultades tiene el modelo para realizar la clasificación. En el eje horizontal se muestra el valor que ha predicho el algoritmo y en el eje vertical el valor real del golpe. Por ejemplo, si un golpe de Volea de Revés (VR → vertical) se ha clasificado como Volea de Revés (R → horizontal) se sumará 1 a dicha casilla. Se puede concluir que cuanto más altos sean los números de la diagonal, mejor será la accuracy del clasificador, ya que serán golpes clasificados como lo que son en realidad.

Se puede observar en el anexo A4.1.1, que los mayores números fuera de la diagonal principal (errores) se dan para golpes que técnicamente son bastante parecidos. En este caso, se ha confundido hasta 17 veces un golpe de revés con un globo de revés.

5.3.1.1.2. Tres capas densamente conectadas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 120)	0
dense_18 (Dense)	(None, 2000)	242000
dense_19 (Dense)	(None, 1000)	2001000
dense_20 (Dense)	(None, 13)	13013

Total params: 2,256,013

Trainable params: 2,256,013

Non-trainable params: 0

La primera y segunda capa del modelo están formadas por una función de activación ReLU y un total de 2000 y 1000 neuronas respectivamente, mientras que la tercera la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **79.864%** (+/-0.573).

5.3.1.1.3. Resumen redes densas

A modo resumen, en la Tabla 5, se muestran los resultados obtenidos para los distintos modelos de redes densas, obteniendo un mejor resultado en la configuración de tres capas.

Tabla 5. Resumen resultados redes densas dominio frecuencial incluyendo deportistas zurdos

	Dos capas densas	Tres capas densas
Accuracy medio (%)	73.946	79.864
Accuracy máximo (%)	74.511	80.437

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 41:

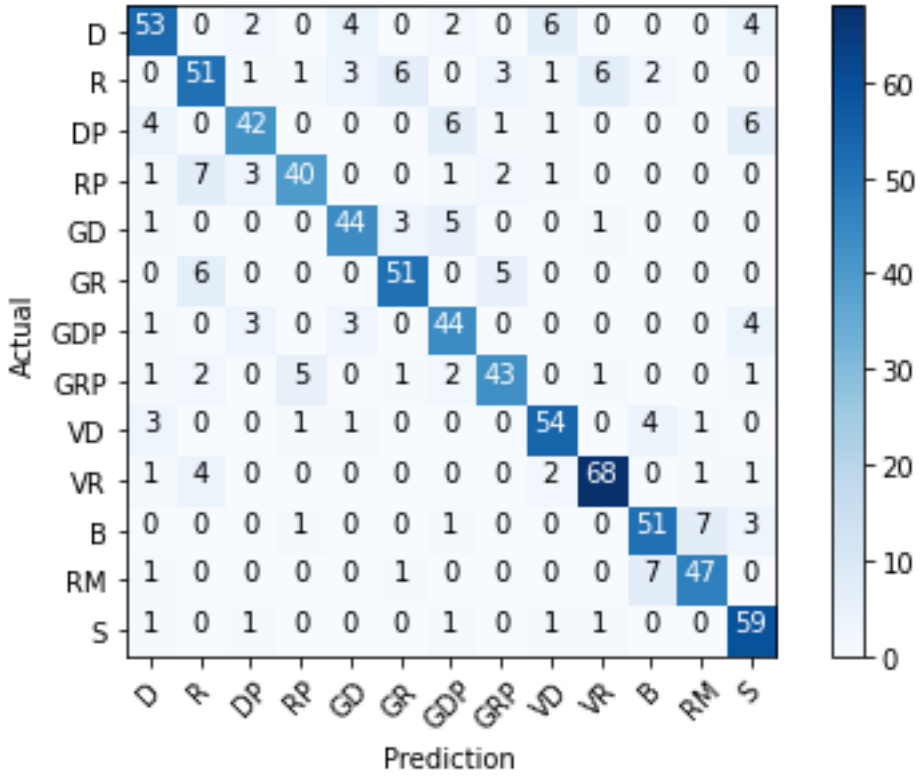


Figura 41. Matriz confusión mejor configuración redes densas en el dominio frecuencial incluyendo deportistas zurdos

Los números más altos fuera de la diagonal principal, es decir, los golpes que más le cuesta distinguir al algoritmo han sido: confundir un remate con una bandeja y viceversa y un revés con un revés con pared, acumulando un total de 7 predicciones erróneas en cada uno. Aunque sean predicciones erróneas demuestran el correcto funcionamiento de los algoritmos, ya que los golpes confundidos presentan una ejecución muy similar, a veces incluso difusa para el ojo humano.

5.3.1.2. Redes neuronales convolucionales 1D

Las redes neuronales densamente conectadas pueden ir precedidas por redes convolucionales 1D. Al igual que en el caso de las redes neuronales densas, se divide el conjunto de datos y se destina un 64% de estos para entrenamiento, un 16% para validación y finalmente un 20% restante para el testeo. Los hiperparámetros que se han de modificar son: el tamaño de kernel, el número de epochs, el batch size, y el número de filtros de la capa convolucional.

Se van a probar distintas configuraciones añadiendo o quitando capas tanto densas como convolucionales.

5.3.1.2.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_65 (Conv1D)	(None, 16, 512)	15872
dropout_45 (Dropout)	(None, 16, 512)	0
max_pooling1d_45 (MaxPoolin g1D)	(None, 8, 512)	0
flatten_89 (Flatten)	(None, 4096)	0
dense_187 (Dense)	(None, 13)	53261

Total params: 69,133
Trainable params: 69,133
Non-trainable params: 0

Posee una primera capa convolucional con 512 filtros y un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **70.801%** (+/-1.234) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.1.2.2. Dos capas convolucionales y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_84 (Conv1D)	(None, 16, 512)	15872
conv1d_85 (Conv1D)	(None, 12, 256)	655616
dropout_57 (Dropout)	(None, 12, 256)	0
max_pooling1d_57 (MaxPoolin g1D)	(None, 6, 256)	0
flatten_101 (Flatten)	(None, 1536)	0
dense_199 (Dense)	(None, 13)	19981

Total params: 691,469
Trainable params: 691,469
Non-trainable params: 0

La primera capa convolucional posee 512 filtros y la segunda 256, ambas con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y una densa con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **72.972%** (+/-0.889) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.1.2.3. Una capa convolucional y dos capas densas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_31 (Conv1D)           (None, 16, 512)           15872
dropout_21 (Dropout)         (None, 16, 512)           0
max_pooling1d_21 (MaxPoolin  (None, 8, 512)            0
g1D)
flatten_65 (Flatten)         (None, 4096)              0
dense_144 (Dense)            (None, 1000)              4097000
dense_145 (Dense)            (None, 13)                13013
=================================================================
Total params: 4,125,885
Trainable params: 4,125,885
Non-trainable params: 0
```

La capa convolucional posee 512 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas, la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **68.015%** (+/-1.367) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_84 (Conv1D)           (None, 16, 512)           15872
conv1d_85 (Conv1D)           (None, 12, 256)           655616
dropout_57 (Dropout)         (None, 12, 256)           0
max_pooling1d_57 (MaxPoolin  (None, 6, 256)            0
g1D)
flatten_101 (Flatten)        (None, 1536)              0
dense_199 (Dense)            (None, 13)                19981
=================================================================
Total params: 691,469
Trainable params: 691,469
Non-trainable params: 0
```

La primera capa convolucional posee 512 filtros y la segunda 256, ambas con un kernel de 5. Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **70.197%** (+/-1.233) y se ha dado para 70 *epochs* y un tamaño de *batch*

size de 30.

5.3.1.2.4. Resumen redes convolucionales 1D

En la Tabla 6, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 6. Resumen resultados redes convolucionales 1D dominio frecuencial incluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	70.801	72.972	68.015	70.197
Accuracy máximo (%)	72.035	73.861	69.382	71.430

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 42:

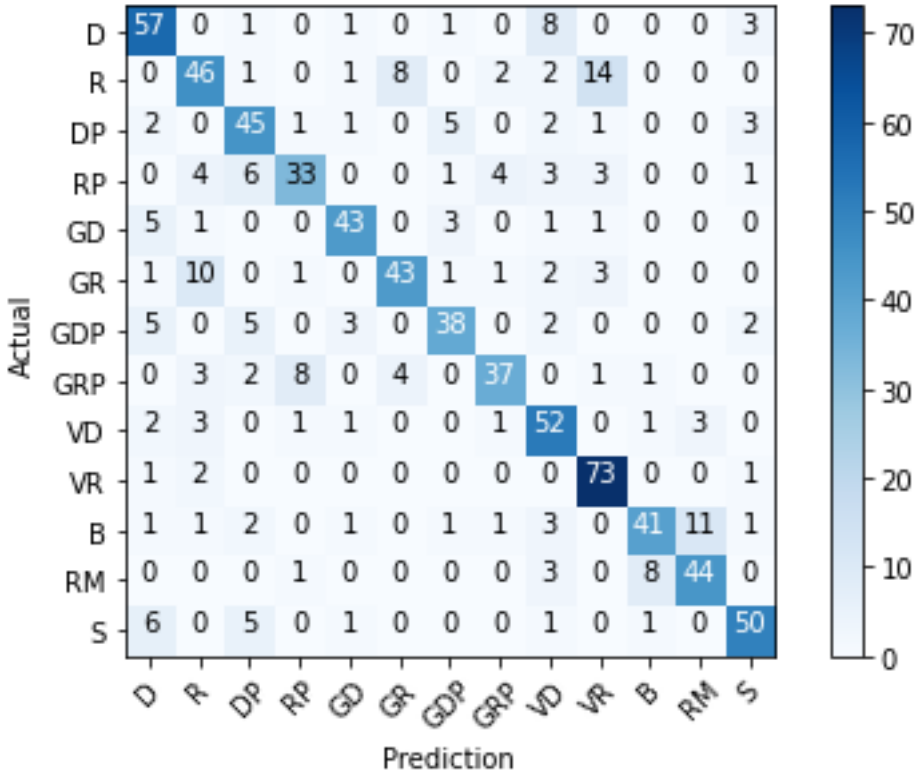


Figura 42. Matriz confusión mejor configuración redes convolucionales 1D en el dominio frecuencial incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con una volea de revés, ya que los ha clasificado mal hasta 14 veces. Los siguientes golpes que más trabajo le ha costado diferenciar han sido la bandeja con el remate, entendible ya que presentan ejecuciones muy similares.

5.3.1.3. Redes neuronales convolucionales 2D

Las redes neuronales densamente conectadas pueden ir precedidas, además de redes convolucionales 1D, de redes convolucionales 2D. Al igual que en el caso de las redes neuronales convolucionales 1D, se divide el conjunto de datos y se destina un 64% de estos para entrenamiento, un 16% para validación y finalmente un 20% restante para el testeo. Los hiperparámetros que se han de modificar son: el tamaño de kernel, el número de epochs, el batch size, y el número de filtros de la capa convolucional. En este caso, se introducirán los datos al algoritmo como si fueran imágenes de tamaño 6x20 (GDL x nº de muestras por golpe) y los filtros las

recorrerán de la forma que se ha explicado en el apartado 3.2.2.3.

Se van a probar distintas configuraciones añadiendo o quitando capas tanto densas como convolucionales.

5.3.1.3.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_215 (Conv2D)          (None, 18, 4, 256)        2560
dropout_142 (Dropout)        (None, 18, 4, 256)        0
max_pooling2d_149 (MaxPooli  (None, 9, 2, 256)         0
ng2D)
flatten_140 (Flatten)        (None, 4608)              0
dense_205 (Dense)            (None, 13)                59917
=================================================================
Total params: 62,477
Trainable params: 62,477
Non-trainable params: 0
```

Posee una primera capa convolucional con 256 filtros y un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **58.940%** (+/-1.685) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_205 (Conv2D)          (None, 18, 4, 256)        2560
conv2d_206 (Conv2D)          (None, 16, 2, 512)        1180160
dropout_133 (Dropout)        (None, 16, 2, 512)        0
max_pooling2d_140 (MaxPooli  (None, 8, 1, 512)         0
ng2D)
flatten_131 (Flatten)        (None, 4096)              0
dense_196 (Dense)            (None, 13)                53261
=================================================================
Total params: 1,235,981
Trainable params: 1,235,981
Non-trainable params: 0
```

Posee una primera capa convolucional con 256 filtros y una segunda con un total de 512, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características

más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **75.216%** (+/-1.132) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_193 (Conv2D)          (None, 18, 4, 256)        2560
dropout_125 (Dropout)        (None, 18, 4, 256)        0
max_pooling2d_132 (MaxPooli  (None, 9, 2, 256)         0
ng2D)
flatten_123 (Flatten)        (None, 4608)              0
dense_184 (Dense)            (None, 1000)              4609000
dense_185 (Dense)            (None, 13)                13013
=================================================================
Total params: 4,624,573
Trainable params: 4,624,573
Non-trainable params: 0
```

La capa convolucional posee 256 filtros con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas, la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **74.057%** (+/-1.132) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_186 (Conv2D)          (None, 18, 4, 256)        2560
conv2d_187 (Conv2D)          (None, 16, 2, 512)        1180160
dropout_121 (Dropout)        (None, 16, 2, 512)        0
max_pooling2d_128 (MaxPooli  (None, 8, 1, 512)         0
ng2D)
flatten_119 (Flatten)        (None, 4096)              0
dense_176 (Dense)            (None, 1000)              4097000
dense_177 (Dense)            (None, 13)                13013
=================================================================
Total params: 5,292,733
Trainable params: 5,292,733
Non-trainable params: 0
```

La primera capa convolucional posee 256 filtros y la segunda 512, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la

primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **75.413%** (+/-0.452) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.1.3.2. Resumen redes convolucionales 2D

En la Tabla 7, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 7. Resumen resultados redes convolucionales 2D dominio frecuencial incluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	58.940	75.216	74.057	75.413
Accuracy máximo (%)	60.625	76.348	75.189	75.865

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 43:

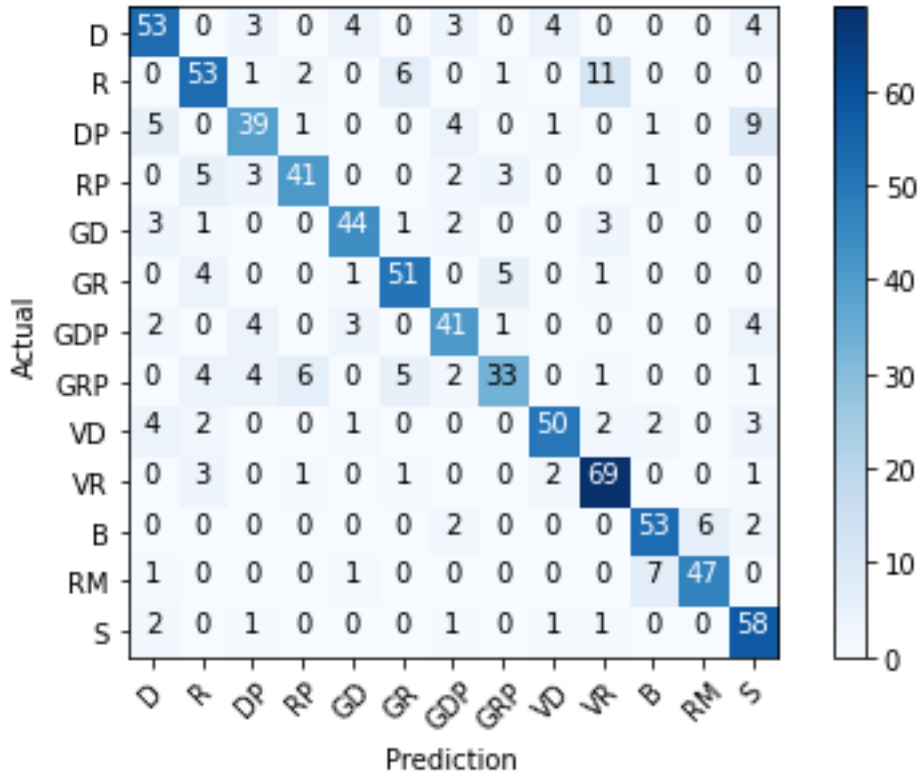


Figura 43. Matriz confusión mejor configuración redes convolucionales 2D en el dominio frecuencial incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con la volea de revés, ya que los ha clasificado mal hasta 11 veces. Además, el remate y la bandeja presentan una técnica fácilmente confundible. Es por esto que el algoritmo los confunde hasta 7 veces.

5.3.1.4. Árbol de decisión

Para encontrar los valores de los hiperparámetros con los que el árbol de decisión obtiene una mayor cantidad de acierto se realiza una búsqueda de rejilla con los valores mostrados en la Tabla 8. El 70% del conjunto de datos se destina para el entrenamiento, mientras que el 30% restante se utiliza para el testeo.

Tabla 8. Hiperparámetros para árbol de decisión.

Max_depth	[1, 10, 20, 30, 40]
Min_samples_split	[2, 4, 8, 10, 20, 100]
Min_samples_leaf	[1, 2, 3, 4, 5, 6, 10]
Criterion	[entropy, gini]

La mayor accuracy obtenida es del **44.762%** (+/-0.165) y se ha dado para una *max_depth* de 40, un *min_samples_split* de 8, un *min_samples*_leaf de 10 y un *criterion* de Gini.

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la siguiente Figura 44:

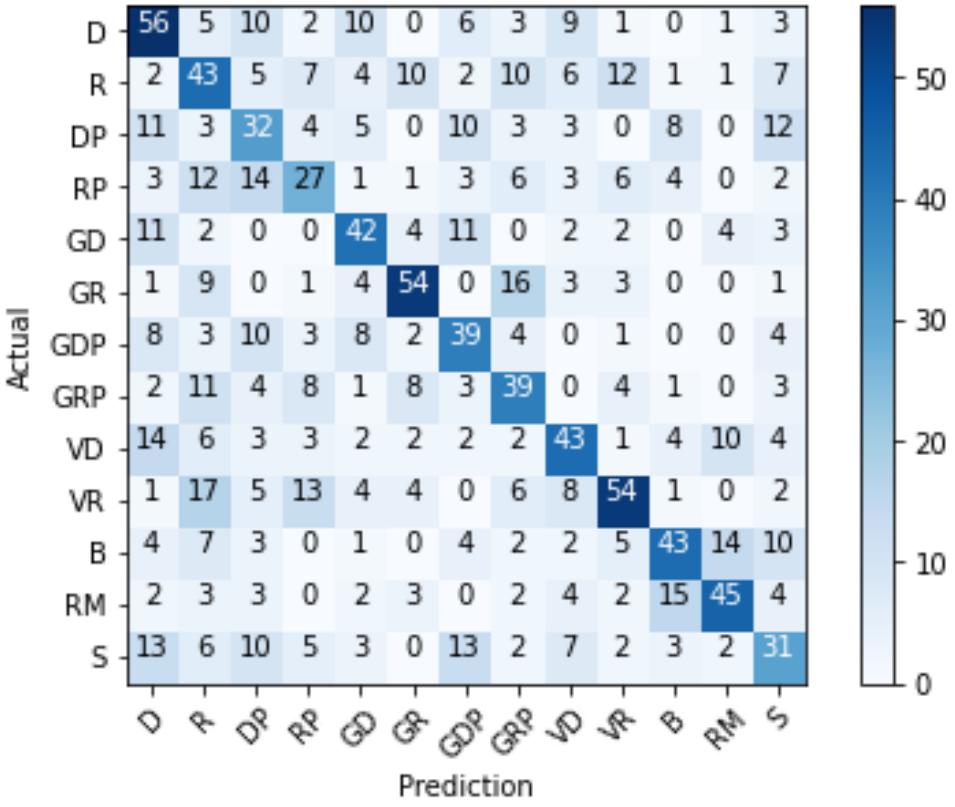


Figura 44. Matriz confusión mejor configuración árbol de decisión en el dominio frecuencial incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el remate con una bandeja, ya que los ha clasificado mal hasta 16 veces y la volea de revés con el revés confundiéndolos hasta 17 veces. Se puede observar el correcto funcionamiento del clasificador, puesto que confunde los golpes que son técnicamente similares, en los que no se parecen nada como el remate con el globo de revés con pared, no los confunde ninguna vez.

5.3.1.5. SVM

Los hiperparámetros que se han modificado con el fin de obtener una mejor accuracy han sido la regularización (C) y el filtro kernel. Al igual que en el caso anterior, el conjunto de datos se va a dividir en el 70% y el 30%, siendo el primer porcentaje el destinado para el entrenamiento y el segundo para el test.

Tabla 9. Resultados SVM en dominio frecuencial incluyendo deportistas zurdos, en función de kernel y C

		Regularización ©								
		0.01	0.1	0.5	1	2	10	12	20	100
Filtro kernel	Lineal	67.76	64.23	63.90	63.24	63.57	63.40	63.49	63.24	62.75
	Polinómico	28.78	58.31	69.49	71.46	73.85	76.48	76.56	77.88	75.90
	Radial	11.02	48.85	64.14	69.33	72.94	80.18	80.76	80.26	79.69
	Sigmoide	9.46	17.52	17.11	12.83	10.44	8.96	8.88	8.63	8.96

La mayor accuracy se ha dado para el filtro kernel de tipo radial y una C de 12, obteniendo un **80.76%** de aciertos.

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la siguiente Figura 45:

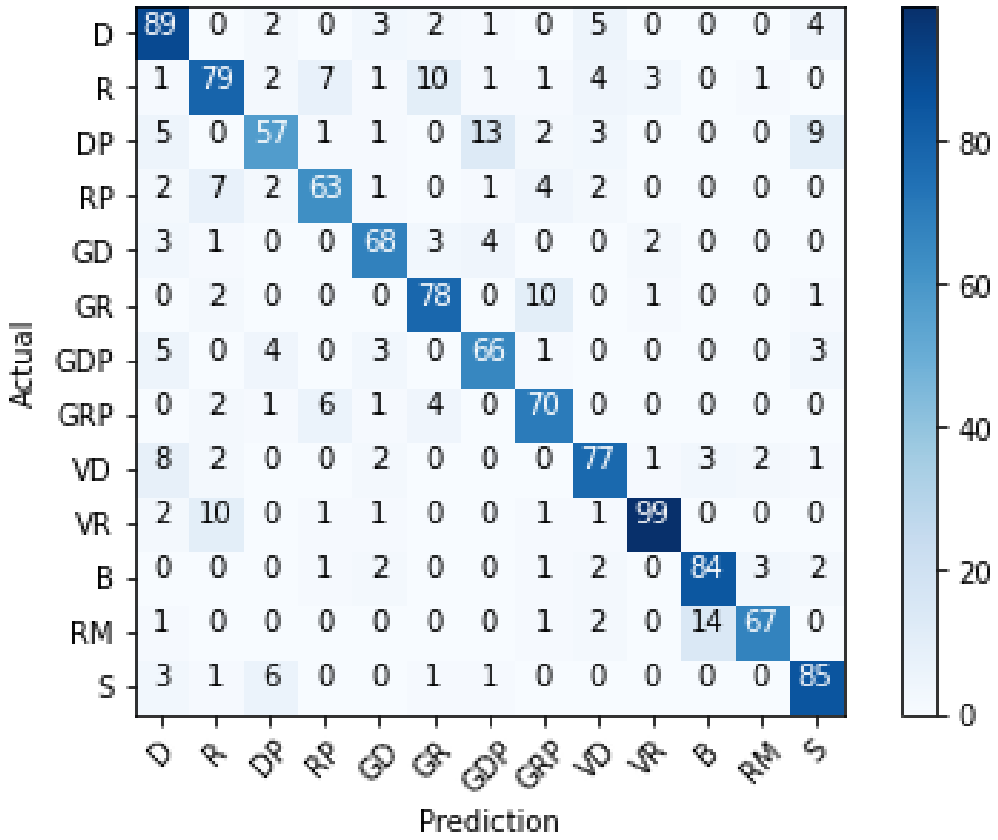


Figura 45. Matriz confusión mejor configuración en SVM en el dominio frecuencial incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el remate con una bandeja, ya que los ha clasificado mal hasta 14 veces. Los siguientes golpes que más trabajo le ha costado diferenciar han sido la derecha con pared y el globo de derecha con pared, entendible ya que presentan ejecuciones muy similares.

5.3.1.6. KNN

El hiperparámetro a modificar en este algoritmo es el valor de k, es decir, la cantidad de vecinos más próximos que se tienen en cuenta para la clasificación. El 70% de los datos se destinarán para el entrenamiento y el 30% restante para el testeo.

Se ha probado para distintos valores de k, desde el 1 (tomando tan solo el vecino más cercano) hasta 30. Se puede observar que se han tomado tanto valores pares como impares, ya que se trata de una clasificación no binaria.

Tabla 10. Resultados KNN en dominio frecuencial incluyendo deportistas zurdos, en función de k

k	1	2	3	4	5	6	7	8	9
Accuracy (%)	79.11	73.77	76.89	76.40	76.07	75.08	74.92	73.19	74.18
k	10	11	12	13	14	15	20	25	30
Accuracy (%)	73.03	73.19	72.70	71.71	70.72	70.89	69.08	67.93	66.53

El mejor resultado se da para k = 1 con un accuracy del **79.11 %**.

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la siguiente Figura 46:

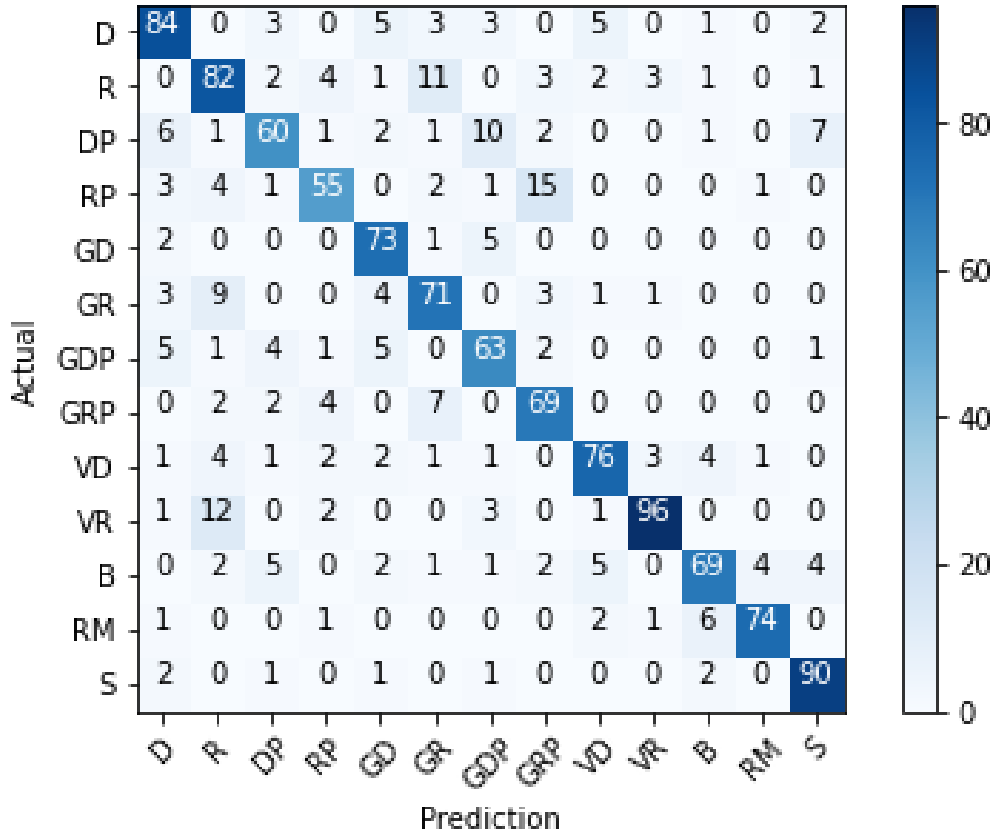


Figura 46. Matriz confusión mejor configuración KNN en el dominio frecuencial incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con pared con globo de revés con pared, ya que los ha clasificado mal hasta 15 veces. Los siguientes golpes que más trabajo le ha costado diferenciar han sido el revés con la volea de revés, ya que presentan ejecuciones muy similares y era de esperar.

5.3.2. Jugadores exclusivamente diestros

En los experimentos realizados en esta sección se omitieron de la base de datos los golpes de los deportistas zurdos, con el fin de observar si existen diferencias notables.

5.3.2.1. Redes neuronales densamente conectadas

5.3.2.1.1. Dos capas densamente conectadas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_110 (Flatten)        (None, 120)               0
dense_214 (Dense)            (None, 2000)              242000
dense_215 (Dense)            (None, 13)                26013
=================================================================
Total params: 268,013
Trainable params: 268,013
Non-trainable params: 0
```

La primera capa del modelo está formada por una función de activación ReLU y un total de 2000 neuronas, mientras que la segunda la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **72.473%** (+/-0.689).

Tres capas densamente conectadas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_118 (Flatten)        (None, 120)               0
dense_234 (Dense)            (None, 2000)              242000
dense_235 (Dense)            (None, 1000)              2001000
dense_236 (Dense)            (None, 13)                13013
=================================================================
Total params: 2,256,013
Trainable params: 2,256,013
Non-trainable params: 0
```

La primera y segunda capa del modelo están formadas por una función de activación ReLU y un total de 2000 y 1000 neuronas respectivamente, mientras que la segunda la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **78.498%** (+/-0.396).

5.3.2.1.2. Resumen redes densas

En la Tabla 11, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas.

Tabla 11. Resumen resultados redes densas dominio frecuencial excluyendo deportistas zurdos

	Dos capas densas	Tres capas densas
Accuracy medio (%)	72.473	78.498
Accuracy máximo (%)	73.162	78.894

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 47:

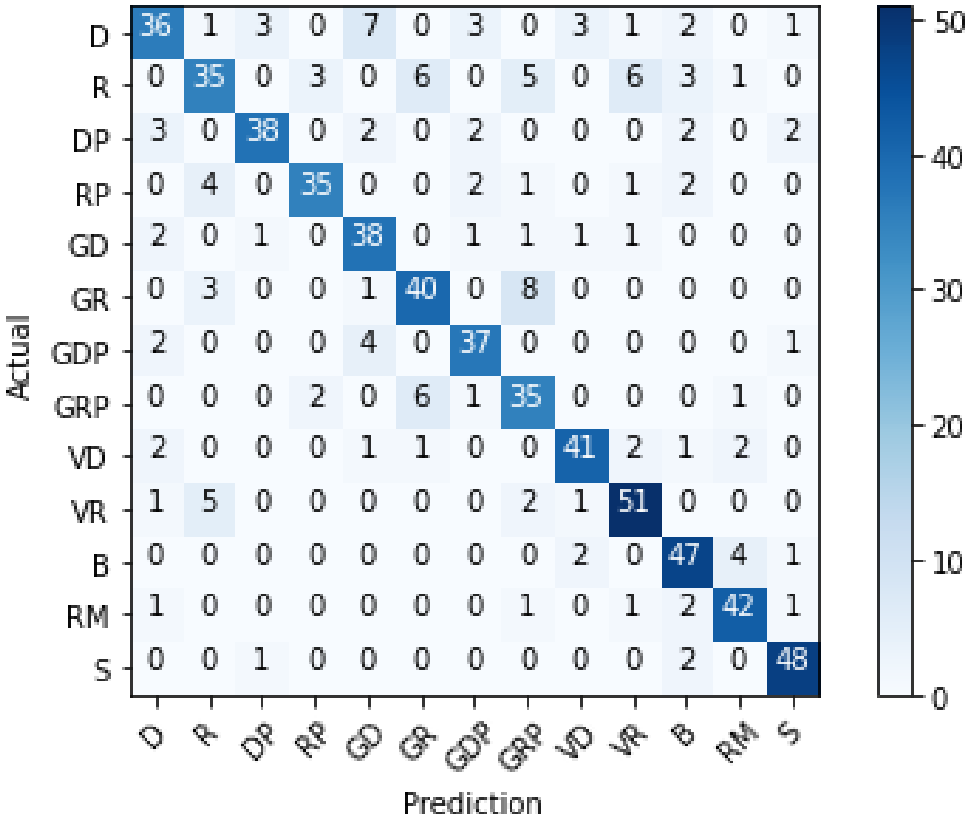


Figura 47. Matriz confusión mejor configuración redes densas en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el globo de revés con pared con globo de revés con pared, ya que los ha clasificado mal hasta 8 veces. Además, presenta dificultades para diferenciar la derecha del globo de derecha, ya que la técnica de ejecución es bastante parecida. Movimientos como el remate con el globo de revés, no tienen nada que ver es por eso que el algoritmo no presenta problemas para su diferenciación.

5.3.2.2. Redes neuronales convolucionales 1D

5.3.2.2.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_115 (Conv1D)	(None, 16, 512)	15872
dropout_75 (Dropout)	(None, 16, 512)	0
max_pooling1d_75 (MaxPooling1D)	(None, 8, 512)	0
flatten_140 (Flatten)	(None, 4096)	0
dense_280 (Dense)	(None, 13)	53261

Total params: 69,133
Trainable params: 69,133
Non-trainable params: 0

La capa convolucional posee 512 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y una densas con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **71.821%** (+/-1.397) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_115 (Conv1D)	(None, 16, 512)	15872
dropout_75 (Dropout)	(None, 16, 512)	0
max_pooling1d_75 (MaxPooling1D)	(None, 8, 512)	0
flatten_140 (Flatten)	(None, 4096)	0
dense_280 (Dense)	(None, 13)	53261

Total params: 69,133
Trainable params: 69,133
Non-trainable params: 0

Posee dos capas convolucionales, la primera con 128 filtros y la segunda con 256, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **87.557%** (+/-1.191) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_148 (Conv1D)	(None, 16, 512)	15872
dropout_98 (Dropout)	(None, 16, 512)	0
max_pooling1d_98 (MaxPoolin g1D)	(None, 8, 512)	0
flatten_163 (Flatten)	(None, 4096)	0
dense_311 (Dense)	(None, 1000)	4097000
dense_312 (Dense)	(None, 13)	13013

Total params: 4,125,885
Trainable params: 4,125,885
Non-trainable params: 0

La capa convolucional posee 512 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y una función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **67.769%** (+/-2.217) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_98 (Conv1D)	(None, 16, 512)	15872
conv1d_99 (Conv1D)	(None, 12, 256)	655616
dropout_64 (Dropout)	(None, 12, 256)	0
max_pooling1d_64 (MaxPoolin g1D)	(None, 6, 256)	0
flatten_129 (Flatten)	(None, 1536)	0
dense_263 (Dense)	(None, 1000)	1537000
dense_264 (Dense)	(None, 13)	13013

Total params: 2,221,501
Trainable params: 2,221,501
Non-trainable params: 0

La primera capa convolucional posee 512 filtros y la segunda un total de 256, aunque ambas con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y una función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **69.181%** (+/-2.196) y se ha dado para 70 *epochs* y

un tamaño de *batch size* de 30.

5.3.2.2.2. Resumen redes convolucionales 1D

En la Tabla 12, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 12. Resumen resultados redes convolucionales 1D dominio frecuencial excluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	71.821	72.489	67.769	69.181
Accuracy máximo (%)	73.218	73.753	69.986	71.377

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 48:

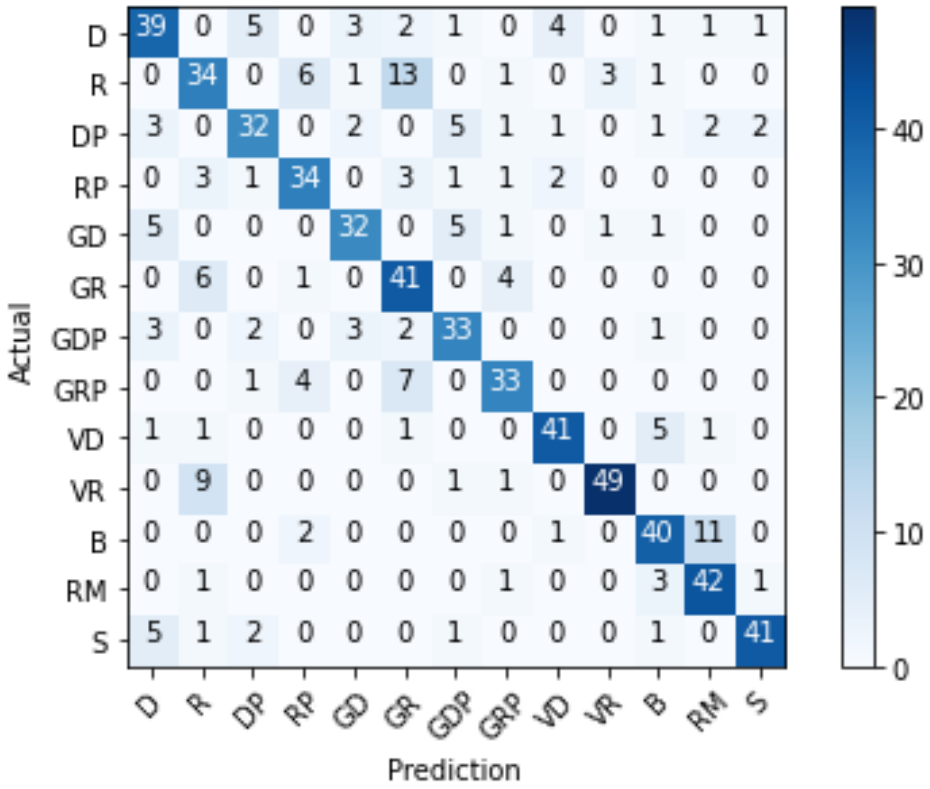


Figura 48. Matriz confusión mejor configuración redes convolucionales 1D en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con el globo de revés, ya que los ha clasificado mal hasta 13 veces. Además, el remate y la bandeja presentan una técnica fácilmente confundible. Es por esto que el algoritmo los confunde hasta 11 veces.

5.3.2.3. Redes neuronales convolucionales 2D

5.3.2.3.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_179 (Conv2D)          (None, 18, 4, 256)        2560
dropout_116 (Dropout)        (None, 18, 4, 256)        0
max_pooling2d_123 (MaxPooli  (None, 9, 2, 256)         0
ng2D)
flatten_114 (Flatten)        (None, 4608)              0
dense_169 (Dense)            (None, 13)                59917
=================================================================
Total params: 62,477
Trainable params: 62,477
Non-trainable params: 0
```

Posee una primera capa convolucional con 256 filtros y un kernel de (3,3). Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **66.950%** (+/- 1.026) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_173 (Conv2D)          (None, 18, 4, 256)        2560
conv2d_174 (Conv2D)          (None, 16, 2, 512)        1180160
dropout_112 (Dropout)        (None, 16, 2, 512)        0
max_pooling2d_119 (MaxPooli  (None, 8, 1, 512)         0
ng2D)
flatten_110 (Flatten)        (None, 4096)              0
dense_165 (Dense)            (None, 13)                53261
=================================================================
Total params: 1,235,981
Trainable params: 1,235,981
Non-trainable params: 0
```

Posee dos capas convolucionales, la primera con 256 filtros y la segunda con 512, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy

logrado es **77.329%** (+/-0.967) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.2.3.2. Una capa convolucional y dos capas densas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_164 (Conv2D)          (None, 18, 4, 256)        2560
dropout_106 (Dropout)        (None, 18, 4, 256)        0
max_pooling2d_113 (MaxPooli  (None, 9, 2, 256)         0
ng2D)
flatten_104 (Flatten)        (None, 4608)              0
dense_156 (Dense)            (None, 1000)              4609000
dense_157 (Dense)            (None, 13)                13013
=================================================================
Total params: 4,624,573
Trainable params: 4,624,573
Non-trainable params: 0
```

La capa convolucional posee 256 filtros con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas, la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **74.962%** (+/-0.974) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.2.3.3. Dos capas convolucionales y dos capas densas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

La primera capa convolucional posee 256 filtros y la segunda 512, ambas con un kernel de 5. Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_158 (Conv2D)          (None, 18, 4, 256)        2560
conv2d_159 (Conv2D)          (None, 16, 2, 512)        1180160
dropout_102 (Dropout)        (None, 16, 2, 512)        0
max_pooling2d_109 (MaxPooli  (None, 8, 1, 512)         0
ng2D)
flatten_100 (Flatten)        (None, 4096)              0
dense_148 (Dense)            (None, 1000)              4097000
dense_149 (Dense)            (None, 13)
=================================================================
Total params: 5,292,733
Trainable params: 5,292,733
```

softmax. El mejor accuracy logrado es **76.024%** (+/-0.955) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.3.2.3.4. Resumen redes convolucionales 2D

En la Tabla 13, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 13. Resumen resultados redes convolucionales 2D dominio frecuencial excluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	66.950	77.329	74.962	76.024
Accuracy máximo (%)	67.976	78.296	75.936	76.979

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 49:

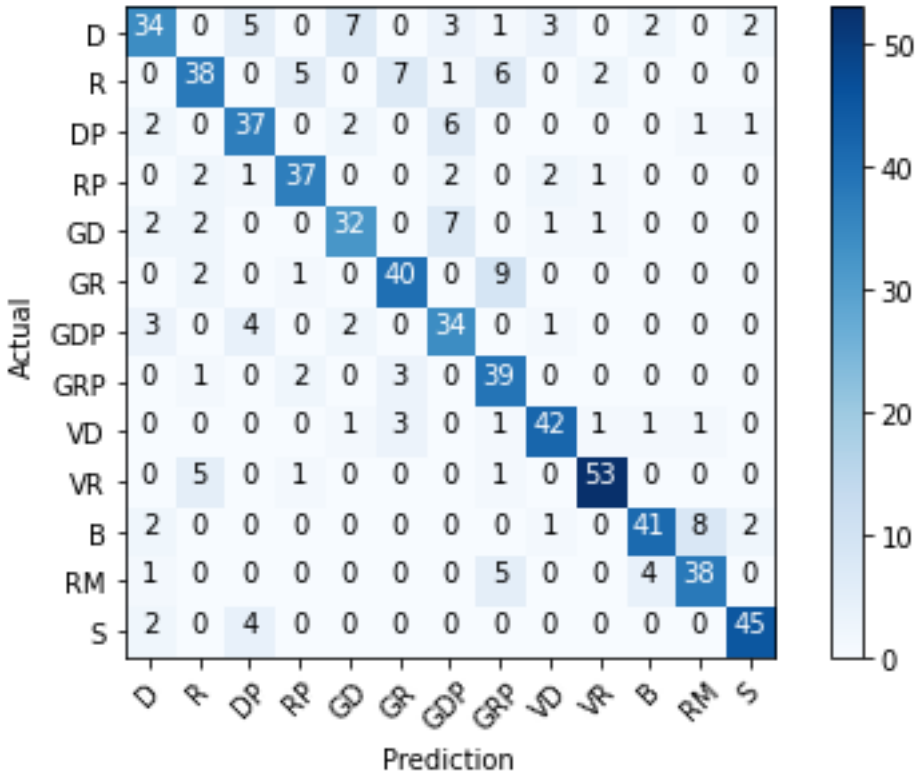


Figura 49. Matriz confusión mejor configuración redes convolucionales 2D en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el globo de revés con el globo de revés con pared, ya que los ha clasificado mal hasta 9 veces. Además, el remate y la bandeja presentan una técnica fácilmente confundible. Es por esto que el algoritmo los confunde hasta 8 veces.

5.3.2.4. Árbol de decisión

Para encontrar los valores de los hiperparámetros con los que el árbol de decisión obtiene una mayor cantidad de aciertos se realiza la misma búsqueda de rejilla que en el apartado anterior, con los valores mostrados en la Tabla 8.

La mayor accuracy obtenida es del **47.128%** (+/-0.413) y se ha dado para una *max_depth* de 30, un *min_samples_split* de 10, un *min_samples_*leaf de 5 y un *criterion* de Gini. La matriz de confusión para este caso se puede observar en la siguiente Figura 50:

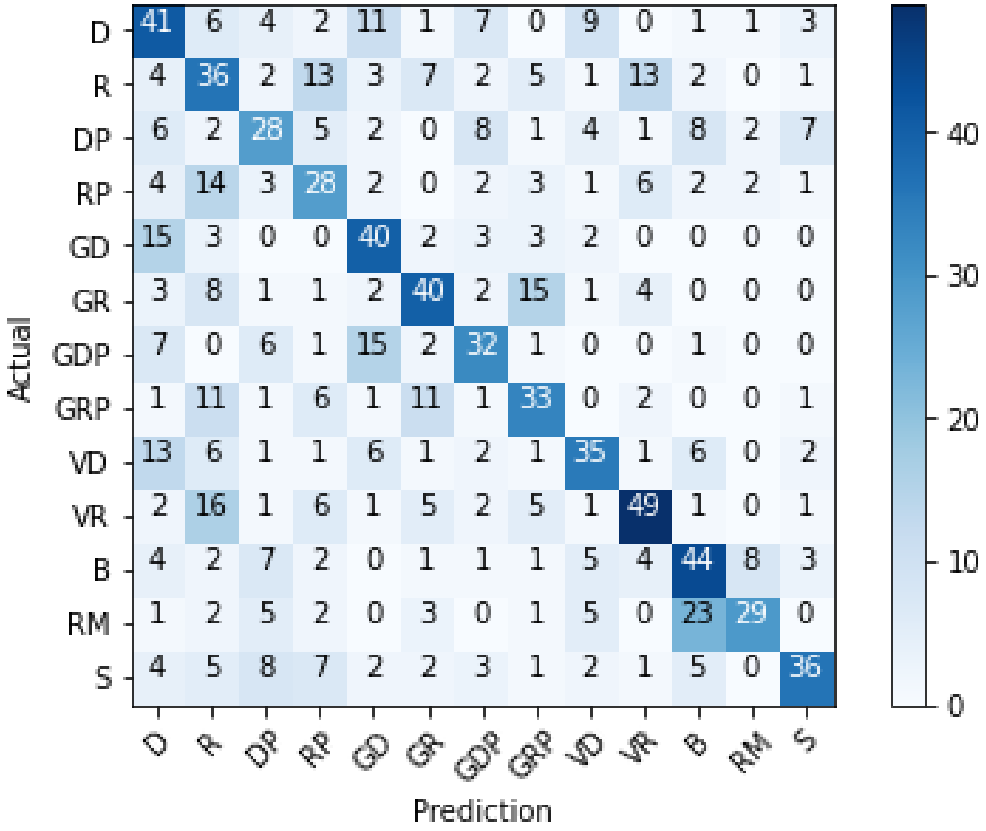


Figura 50. Matriz confusión mejor configuración árbol de decisión en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el remate con la bandeja, ya que los ha clasificado mal hasta 23 veces. Otros golpes que le cuesta diferenciar son la volea de revés con el revés, el globo de derecha con la derecha, el globo de derecha con el globo de derecha con pared y el globo de revés con el globo de revés con pared.

5.3.2.5. SVM

Los hiperparámetros que se han modificado con el fin de obtener una mejor accuracy son los mismos que en el apartado anterior: la regularización (C) y el filtro kernel.

Tabla 14. Resultados SVM en dominio frecuencial excluyendo deportistas zurdos, en función de kernel y C

		Regularización ©								
		0.01	0.1	0.5	1	2	10	12	20	100
Filtro kernel	Lineal	67.95	65.22	64.31	64.21	64.00	63.90	63.90	63.90	63.90
	Polinómico	27.60	57.94	68.76	72.50	74.62	78.87	78.56	78.77	77.05
	Radial	9.61	50.56	64.61	70.48	73.51	80.69	80.79	82.10	79.98
	Sigmoide	9.10	18.20	19.92	14.86	11.93	10.41	10.21	9.71	9.61

La mayor accuracy se ha dado para el filtro kernel de tipo radial y una C de 20, obteniendo un **82.10%** de aciertos. La matriz de confusión para este caso se muestra en la siguiente Figura 51:

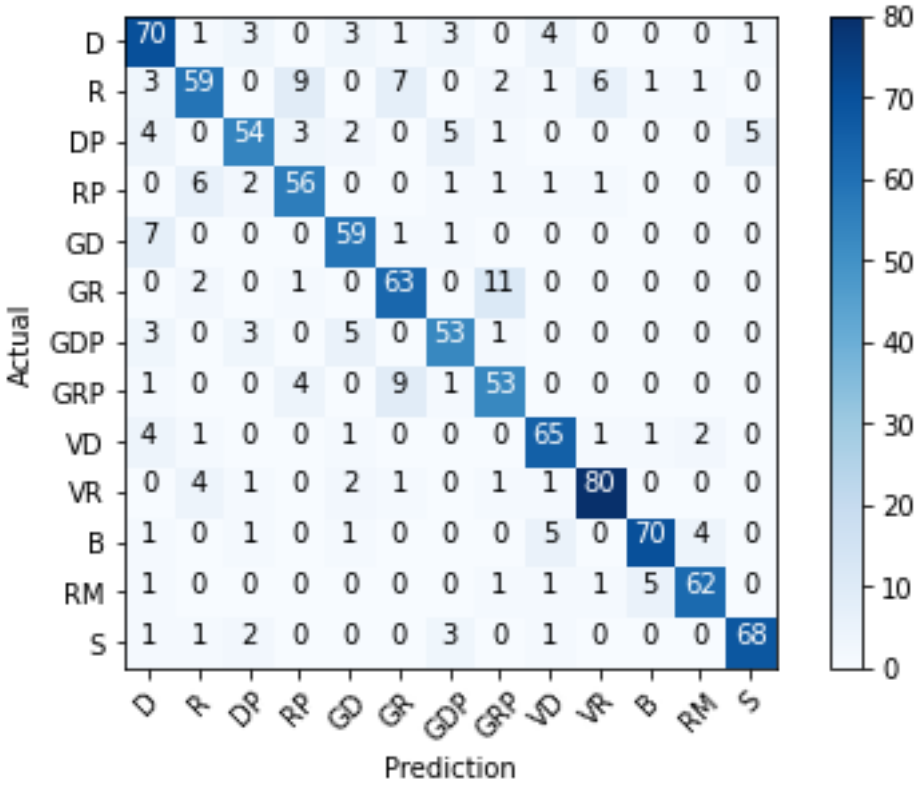


Figura 51. Matriz confusión mejor configuración en SVM en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el globo de revés con pared con globo de revés con pared, ya que los ha clasificado mal hasta 11 veces y el revés con el revés con pared, confundiéndolos un total de 9 veces.

5.3.2.6. KNN

El hiperparámetro a modificar en este algoritmo es el valor de k, es decir, la cantidad de vecinos más próximos que se tienen en cuenta para la clasificación. Se ha probado para distintos valores de k, desde el 1 (tomando tan solo el vecino más cercano) hasta 30.

Tabla 15. Resultados KNN en dominio frecuencial excluyendo deportistas zurdos, en función de k

k	1	2	3	4	5	6	7	8	9
Accuracy (%)	80.59	75.73	76.74	75.53	75.83	74.82	75.23	75.33	74.52
k	10	11	12	13	14	15	20	25	30
Accuracy (%)	74.92	74.82	74.22	74.01	73.41	72.90	68.86	66.13	64.61

El mejor resultado se da para k = 1 con un accuracy del **80.59 %**. La matriz de confusión para este caso se puede observar en la siguiente Figura 52

:

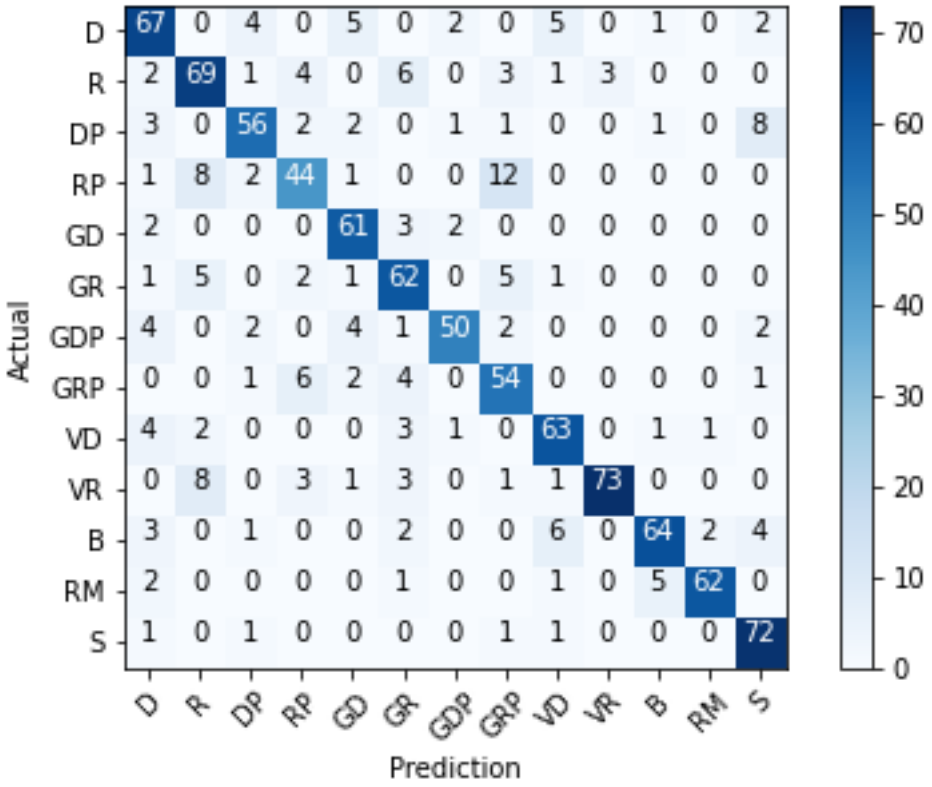


Figura 52. Matriz confusión mejor configuración en KNN en el dominio frecuencial excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con pared con el globo de revés con pared, ya que los ha clasificado mal hasta 12 veces.

5.3.3. Resumen de mejores resultados dominio de la frecuencia

En la Tabla 16, se muestra un resumen de los resultados de los experimentos. En ella se puede observar el mejor accuracy de cada algoritmo en el dominio de la frecuencia, para los casos en los que se han incluido golpes de deportistas zurdos y en los que no.

Tabla 16. Resumen resultados en el dominio frecuencial.

		Con zurdos		Sin zurdos	
		Accuracy medio (%)	**Accuracy máximo (%)**	**Accuracy medio (%)**	**Accuracy máximo (%)**
Redes neuronales densas	Dos capas densas	73.946	74.511	72.473	73.162
	Tres capas densas	79.864	80.437	78.498	78.894
Redes neuronales convolucionales 1D	Una capa convolucional y una capa densa	70.801	72.035	71.821	73.218
	Dos capas convolucionales y una capa densa	72.972	73.861	72.480	73.753
	Una capa convolucional y dos capas densas	68.015	69.382	67.769	69.986
	Dos capas convolucionales y dos capas densas	70.197	71.430	69.181	71.377
Redes neuronales convolucionales 2D	Una capa convolucional y una capa densa	58.940	60.625	66.950	67.976
	Dos capas convolucionales y una capa densa	75.216	76.348	77.329	78.296
	Una capa convolucional y dos capas densas	74.057	75.189	74.962	75.936
	Dos capas convolucionales y dos capas densas	75.413	75.865	76.024	76.979
Árbol de decisión		44.762	44.927	47.128	47.541
SVM		80.760	81.985	82.100	83.781
KNN		79.110	80.052	80.590	87.986

Como conclusión general, para los dos casos a estudiar en el dominio de la frecuencia, el mejor accuracy se consigue con el algoritmo de las máquinas vector soporte. En el caso del entrenamiento de la red con los datos de dos tipos de jugadores, zurdos y diestros, se logran un 80.760% de aciertos de media. En cambio, cuando se entrena a la red con datos exclusivos de jugadores diestros se aumenta el porcentaje de clasificaciones correctas aumenta en un 1.34%, llegando al 83.781%.

En líneas generales, el hecho de introducir jugadores zurdos en la base de datos influye negativamente en los algoritmos KNN, SVM y árbol de decisión en el dominio frecuencial. Los deportistas zurdos, realizan movimientos distintos en ciertos golpeos, por lo que si se utilizan sus datos para entrenar al modelo que clasificará golpes de diestros, lo hará de forma errónea. En cambio, para los modelos que utilizan tanto redes

neuronales densas como convolucionales, les es indiferente con qué tipo de jugadores se entrene el modelo. Se observa que con los datos de jugadores zurdos obtienen mayor accuracy que sin ellos.

5.4. Resultados en el dominio del tiempo

5.4.1. Jugadores zurdos y diestros

5.4.1.1. Redes neuronales densamente conectadas

5.4.1.1.1. Dos capas densamente conectadas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_67 (Flatten)         (None, 240)               0
dense_191 (Dense)            (None, 1000)              241000
dense_192 (Dense)            (None, 13)                13013
=================================================================
Total params: 254,013
Trainable params: 254,013
Non-trainable params: 0
```

La primera capa del modelo está formada por una función de activación ReLU y un total de 1000 neuronas, mientras que la segunda la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **89.075%** (+/-0.387).

Tres capas densamente conectadas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_82 (Flatten)         (None, 240)               0
dense_226 (Dense)            (None, 1000)              241000
dense_227 (Dense)            (None, 500)               500500
dense_228 (Dense)            (None, 13)                6513
=================================================================
Total params: 748,013
Trainable params: 748,013
Non-trainable params: 0
```

La primera y segunda capa del modelo están formadas por una función de activación ReLU y un total de 1000 y 500 neuronas respectivamente, mientras que la tercera la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **89.137%** (+/-0.422).

5.4.1.1.2. Resumen redes densas

En la Tabla 17, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas.

Tabla 17. Resumen resultados redes densas dominio temporal incluyendo deportistas zurdos

	Dos capas densas	Tres capas densas
Accuracy medio (%)	89.075	89.137
Accuracy máximo (%)	89.462	89.559

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la siguiente Figura 53:

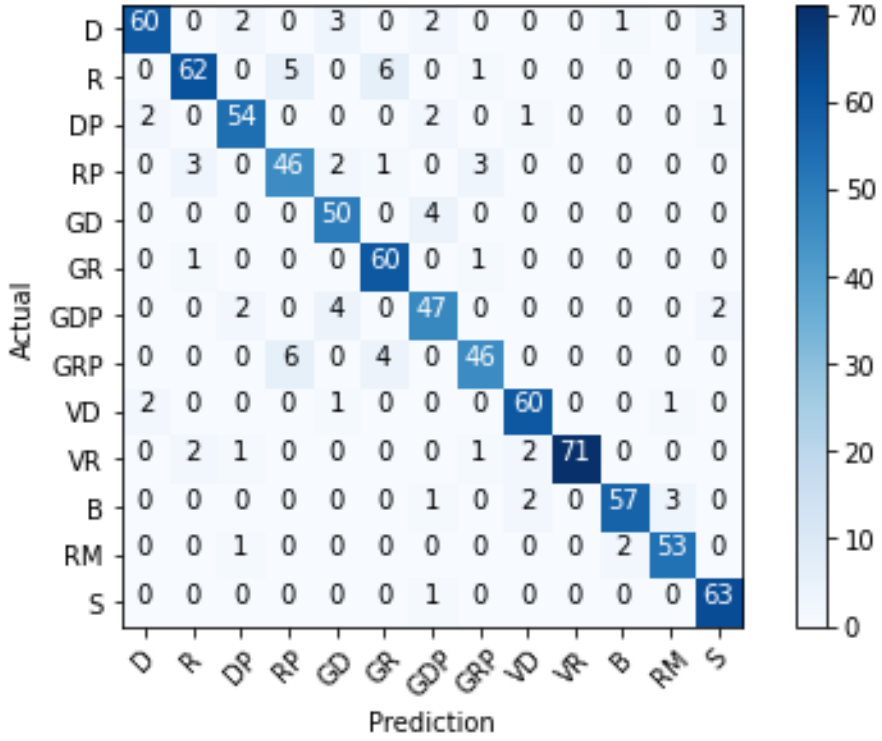


Figura 53. Matriz confusión mejor configuración redes densas en el dominio temporal incluyendo deportistas zurdos

Los números más altos fuera de la diagonal principal, es decir, los golpes que más le cuesta distinguir al algoritmo han sido: confundir un revés con pared con un globo de revés con pared y un revés con un globo de revés, acumulando un total de 6 predicciones erróneas en cada uno.

5.4.1.2. Redes neuronales convolucionales 1D

5.4.1.2.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_140 (Conv1D)          (None, 36, 128)           3968
dropout_90 (Dropout)         (None, 36, 128)           0
max_pooling1d_90 (MaxPoolin  (None, 18, 128)           0
g1D)
flatten_177 (Flatten)        (None, 2304)              0
dense_372 (Dense)            (None, 13)                29965
=================================================================
Total params: 33,933
Trainable params: 33,933
Non-trainable params: 0
```

La capa convolucional posee 128 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y una densas con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **84.303%** (+/-0.979) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 50.

5.4.1.2.2. Dos capas convolucionales y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_109 (Conv1D)          (None, 36, 256)           7936
conv1d_110 (Conv1D)          (None, 32, 128)           163968
dropout_65 (Dropout)         (None, 32, 128)           0
max_pooling1d_65 (MaxPoolin  (None, 16, 128)           0
g1D)
flatten_152 (Flatten)        (None, 2048)              0
dense_337 (Dense)            (None, 13)                26637
=================================================================
Total params: 198,541
Trainable params: 198,541
Non-trainable params: 0
```

La primera capa convolucional posee 256 filtros y la segunda un total de 128, aunque ambas con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y una densa, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **87.904%** (+/-0.674)

y se ha dado para 70 *epochs* y un tamaño de *batch size* de 70.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_129 (Conv1D)	(None, 36, 256)	7936
dropout_79 (Dropout)	(None, 36, 256)	0
max_pooling1d_79 (MaxPoolin g1D)	(None, 18, 256)	0
flatten_166 (Flatten)	(None, 4608)	0
dense_359 (Dense)	(None, 1000)	4609000
dense_360 (Dense)	(None, 13)	13013

Total params: 4,629,949
Trainable params: 4,629,949
Non-trainable params: 0

La capa convolucional posee 256 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y una función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **87.633%** (+/-2.668) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 70.

5.4.1.2.3. Dos capas convolucionales y dos capas densas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_153 (Conv1D)	(None, 36, 128)	3968
conv1d_154 (Conv1D)	(None, 32, 64)	41024
dropout_97 (Dropout)	(None, 32, 64)	0
max_pooling1d_97 (MaxPoolin g1D)	(None, 16, 64)	0
flatten_184 (Flatten)	(None, 1024)	0
dense_385 (Dense)	(None, 1000)	1025000
dense_386 (Dense)	(None, 13)	13013

Total params: 1,083,005
Trainable params: 1,083,005
Non-trainable params: 0

La primera capa convolucional posee 128 filtros y la segunda un total de 64, aunque ambas con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos

densas: la primera con 1000 neuronas y una función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **88.471%** (+/-0.602) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

5.4.1.2.4. Resumen redes convolucionales 1D

En la Tabla 18, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 18. Resumen resultados redes convolucionales 1D dominio temporal incluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	84.303	87.904	87.633	88.471
Accuracy máximo (%)	85.282	88.578	90.301	89.073

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 54:

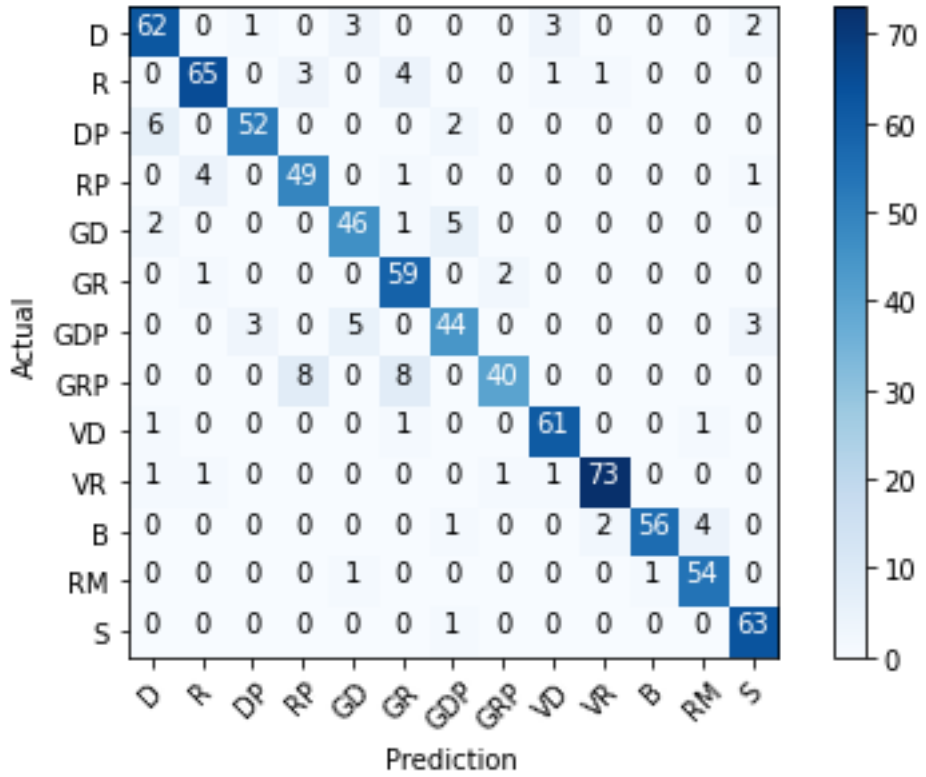


Figura 54. Matriz confusión mejor configuración redes convolucionales 1D en el dominio temporal incluyendo deportistas zurdos

Los números más altos fuera de la diagonal principal, es decir, los golpes que más le cuesta distinguir al algoritmo han sido: confundir un revés con pared con un globo de revés con pared y un globo de revés con pared con un globo de revés, acumulando un total de 8 predicciones erróneas en cada uno.

5.4.1.3. Redes neuronales convolucionales 2D

5.4.1.3.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_142 (Conv2D)          (None, 38, 4, 256)        2560
dropout_92 (Dropout)         (None, 38, 4, 256)        0
max_pooling2d_99 (MaxPoolin  (None, 19, 2, 256)        0
g2D
flatten_90 (Flatten)         (None, 9728)              0
dense_132 (Dense)            (None, 13)                126477
=================================================================
Total params: 129,037
Trainable params: 129,037
Non-trainable params: 0
```

Posee una primera capa convolucional con 512 filtros y un kernel de (3,3). Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **81.578%** (+/- 0.660) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_139 (Conv2D)          (None, 38, 4, 128)        1280
conv2d_140 (Conv2D)          (None, 36, 2, 256)        295168
dropout_90 (Dropout)         (None, 36, 2, 256)        0
max_pooling2d_97 (MaxPoolin  (None, 18, 1, 256)        0
g2D)
flatten_88 (Flatten)         (None, 4608)              0
dense_130 (Dense)            (None, 13)                59917
=================================================================
Total params: 356,365
Trainable params: 356,365
Non-trainable params: 0
```

Posee dos capas convolucionales, la primera con 128 filtros y la segunda con 256, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy

logrado es **85.656%** (+/-1.949) y se ha dado para 100 *epochs* y un tamaño de *batch size* de 30.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_123 (Conv2D)          (None, 38, 4, 128)        1280
dropout_79 (Dropout)         (None, 38, 4, 128)        0
max_pooling2d_86 (MaxPoolin  (None, 19, 2, 128)        0
g2D)
flatten_77 (Flatten)         (None, 4864)              0
dense_113 (Dense)            (None, 1000)              4865000
dense_114 (Dense)            (None, 13)                13013
=================================================================
Total params: 4,879,293
Trainable params: 4,879,293
Non-trainable params: 0
```

La capa convolucional posee 128 filtros con un kernel de (3,3) Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas, la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **86.621%** (+/-1.469) y se ha dado para 100 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_104 (Conv2D)          (None, 38, 4, 128)        1280
conv2d_105 (Conv2D)          (None, 36, 2, 256)        295168
dropout_69 (Dropout)         (None, 36, 2, 256)        0
max_pooling2d_76 (MaxPoolin  (None, 18, 1, 256)        0
g2D)
flatten_67 (Flatten)         (None, 4608)              0
dense_93 (Dense)             (None, 1000)              4609000
dense_94 (Dense)             (None, 13)                13013
=================================================================
Total params: 4,918,461
Trainable params: 4,918,461
Non-trainable params: 0
```

fue la siguiente:

La primera capa convolucional posee 128 filtros y la segunda 256, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas:

la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **88.101%** (+/-1.948) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

5.4.1.3.2. Resumen redes convolucionales 2D

En la Tabla 7, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 19. Resumen resultados redes convolucionales 2D dominio temporal incluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	81.578	85.656	86.621	88.101
Accuracy máximo (%)	82.268	87.605	88.090	90.049

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 55:

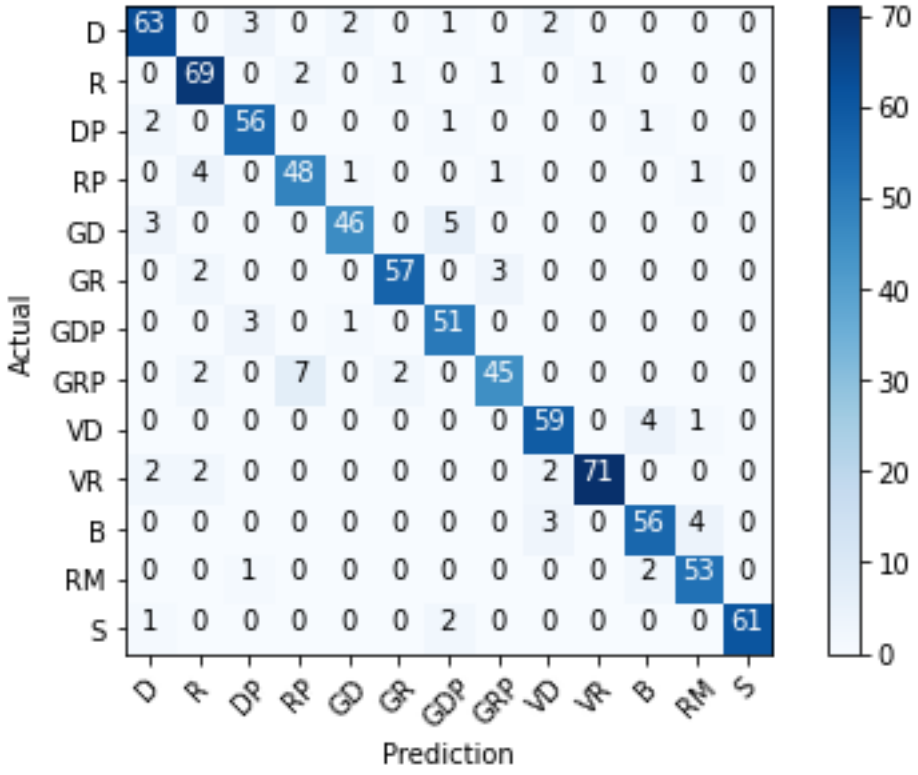


Figura 55. Matriz confusión mejor configuración redes convolucionales 2D en el dominio temporal incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con pared con el globo de revés con pared, ya que los ha clasificado mal hasta 7 veces. Además, el globo de derecha con pared y el globo de derecha presentan una técnica fácilmente confundible. Es por esto que el algoritmo los confunde hasta 5 veces.

5.4.1.4. Árbol de decisión

Para encontrar los valores de los hiperparámetros con los que el árbol de decisión obtiene una mayor cantidad de aciertos se realiza la misma búsqueda de rejilla que en el apartado anterior, con los valores mostrados en la Tabla 8.

La mayor accuracy obtenida es del **53.898%** (+/-0.873) y se ha dado para una *max_depth* de 40, un *min_samples_split* de 2, un *min_samples_leaf* de 1 y un *criterion* de entropy. La matriz de confusión para este

caso se puede observar en la siguiente Figura 56:

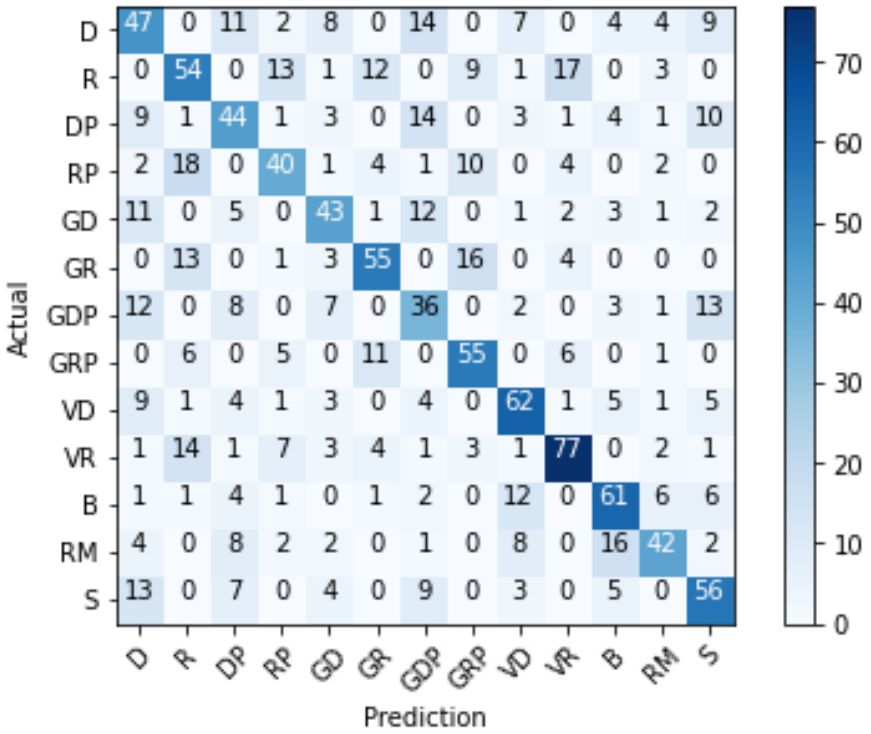


Figura 56. Matriz confusión mejor configuración árbol de decisión en el dominio temporal incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con pared con el revés, ya que los ha clasificado mal hasta 18 veces. También tiene dificultades con la volea de revés y el revés, así como la derecha con el revés y el globo de revés con pared, cuyas técnicas resultan muy similares.

5.4.1.5. SVM

Los hiperparámetros que se han modificado con el fin de obtener una mejor accuracy son los mismos que en el apartado anterior: la regularización (C) y el filtro kernel.

Tabla 20. Resultados SVM en dominio temporal incluyendo deportistas zurdos, en función de kernel y C

		Regularización ©								
		0.01	0.1	0.5	1	2	10	12	20	100
Filtro kernel	Lineal	60.53	57.57	54.61	55.02	53.21	52.63	52.63	52.63	52.63
	Polinómico	10.69	47.45	71.05	77.55	81.91	85.28	85.28	85.12	85.28
	Radial	11.18	47.78	76.89	84.29	87.34	90.54	90.38	89.97	89.64
	Sigmoide	13.32	24.10	26.32	22.04	17.85	14.56	14.23	13.98	14.06

La mayor accuracy se ha dado para el filtro kernel de tipo radial y una C de 10, obteniendo un **90.54%** de aciertos. La matriz de confusión para este caso se puede observar en la siguiente Figura 57:

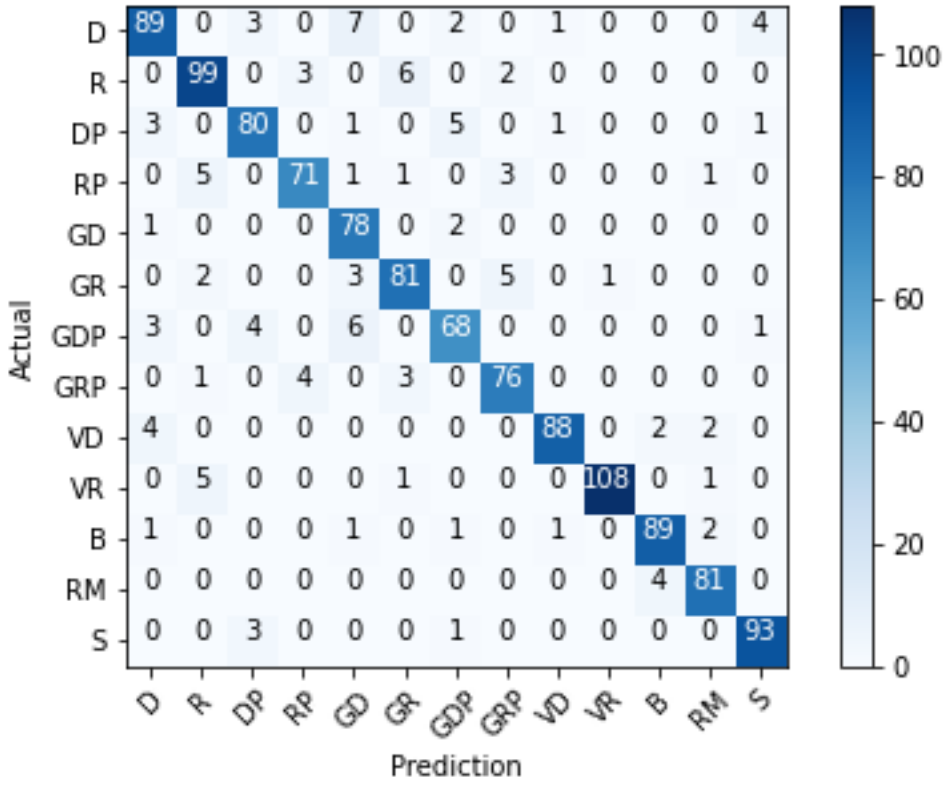


Figura 57. Matriz confusión mejor configuración en SVM en el dominio temporal incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido la derecha con el globo de derecha, ya que los ha clasificado mal hasta 7 veces.

5.4.1.6. KNN

valor de k, es decir, la cantidad de vecinos más próximos que se tienen en cuenta para la clasificación. Se ha probado para distintos valores de k, desde el 1 (tomando tan solo el vecino más cercano) hasta 30

Tabla 21. Resultados KNN en dominio temporal incluyendo deportistas zurdos, en función de k

k	1	2	3	4	5	6	7	8	9
Accuracy (%)	81.83	76.64	78.78	77.80	77.06	77.47	76.48	75.41	73.77
k	10	11	12	13	14	15	20	25	30
Accuracy (%)	72.62	71.05	70.64	69.16	68.26	67.52	64.88	63.57	61.76

El mejor resultado se da para k = 1 con un accuracy del **81.83 %**. La matriz de confusión para este caso se puede observar en la siguiente Figura 58:

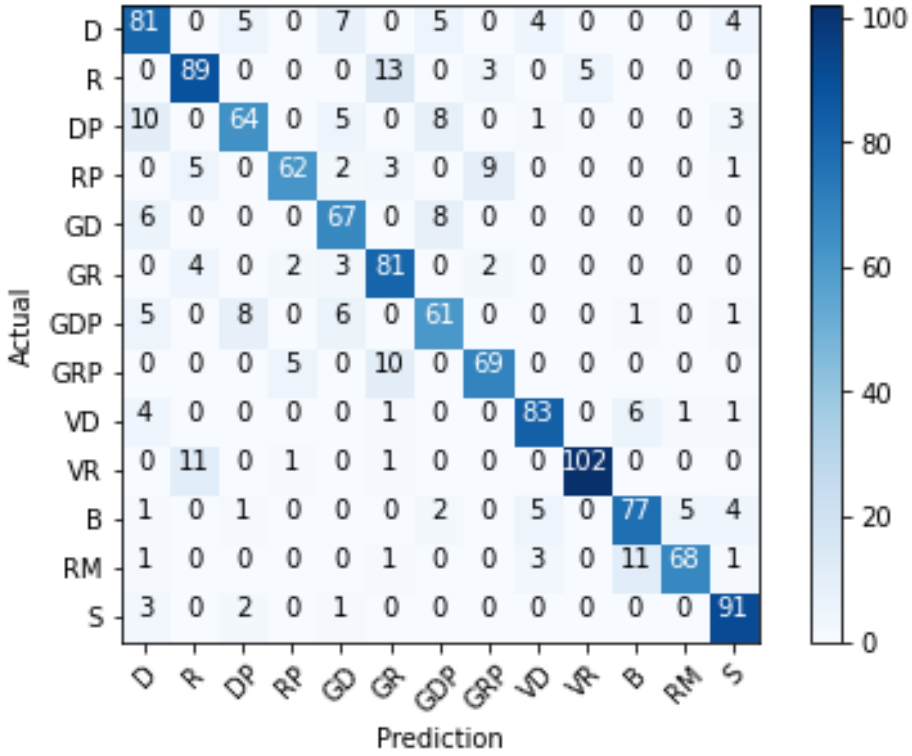


Figura 58. Matriz confusión mejor configuración en KNN en el dominio temporal incluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el reves con el globo de revés, ya que los ha clasificado mal hasta 13 veces. También presenta dificultades para distinguir el revés con la volea de revés y el remate con la bandeja, confundiéndolos hasta 11 veces.

5.4.2. Jugadores exclusivamente diestros

5.4.2.1. Redes neuronales densamente conectadas

5.4.2.1.1. Dos capas densamente conectadas

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_50 (Flatten)         (None, 240)               0
dense_150 (Dense)            (None, 1000)              241000
dense_151 (Dense)            (None, 13)                13013
=================================================================
Total params: 254,013
Trainable params: 254,013
Non-trainable params: 0
```

La primera capa del modelo está formada por una función de activación ReLU y un total de 1000 neuronas, mientras que la segunda la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una

accuracy del **90.212%** (+/-0.522).

Tres capas densamente conectadas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten_45 (Flatten)         (None, 240)               0
dense_135 (Dense)            (None, 1000)              241000
dense_136 (Dense)            (None, 500)               500500
dense_137 (Dense)            (None, 13)                6513
=================================================================
Total params: 748,013
Trainable params: 748,013
Non-trainable params: 0
```

La primera y segunda capa del modelo están formadas por una función de activación ReLU y un total de 1000 y 500 neuronas respectivamente, mientras que la tercera la forma una función de activación de tipo *softmax* y un total de 13 neuronas. El número de epochs que mejora la accuracy es 70 y el tamaño de *batch size*, 30. Con estos hiperparámetros se obtiene una accuracy del **91.062%** (+/-0.467).

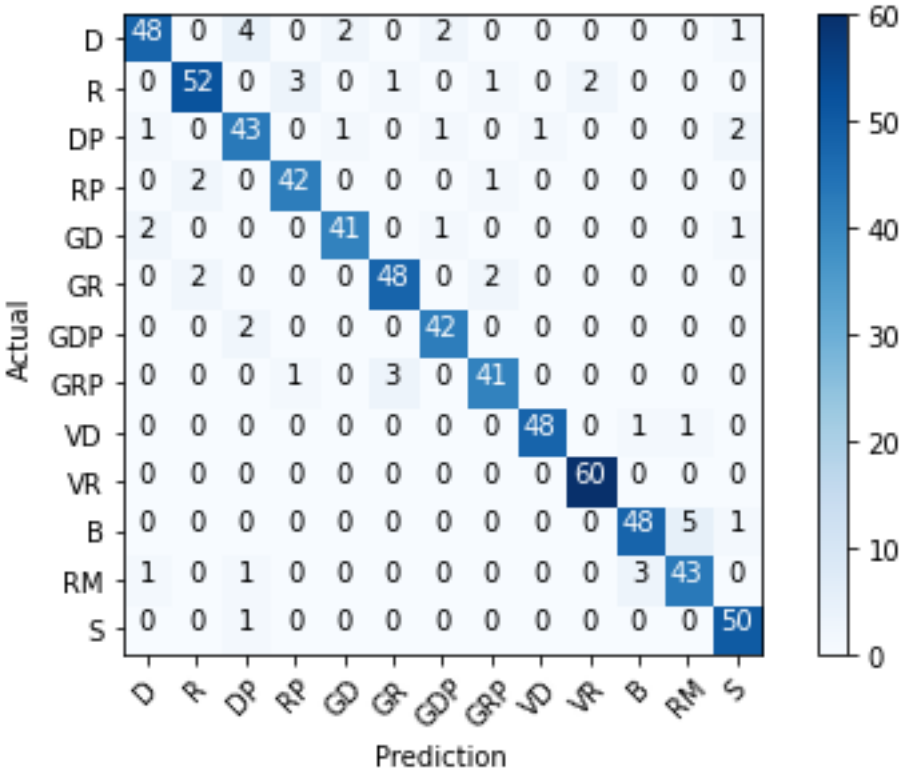
5.4.2.1.2. Resumen redes densas

En la Tabla 22, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas.

Tabla 22. Resumen resultados redes densas dominio temporal excluyendo deportistas zurdos

	Dos capas densas	Tres capas densas
Accuracy medio (%)	90.212	91.062
Accuracy máximo (%)	90.734	91.529

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la siguiente Figrua 59:



Figrua 59. Matriz confusión mejor configuración redes densas en el dominio temporal excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el remate con la bandeja, ya que los ha clasificado mal hasta 5 veces. También, es común confundir la derecha con la derecha con pared. Resultado esperado ya que presentan técnicas muy similares.

5.4.2.2. Redes neuronales convolucionales 1D

5.4.2.2.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_68 (Conv1D)	(None, 36, 128)	3968
dropout_38 (Dropout)	(None, 36, 128)	0
max_pooling1d_38 (MaxPoolin g1D)	(None, 18, 128)	0
flatten_125 (Flatten)	(None, 2304)	0
dense_299 (Dense)	(None, 13)	29965

Total params: 33,933
Trainable params: 33,933
Non-trainable params: 0

La capa convolucional posee 128 filtros con un kernel de 5. Seguidamente se añade una capa *dropout* y otra

maxpooling. A continuación, se añaden una capa *flatten* y una densas con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **88.103%** (+/-0.699) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 50.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_48 (Conv1D)	(None, 36, 128)	3968
conv1d_49 (Conv1D)	(None, 32, 64)	41024
dropout_24 (Dropout)	(None, 32, 64)	0
max_pooling1d_24 (MaxPoolin g1D)	(None, 16, 64)	0
flatten_111 (Flatten)	(None, 1024)	0
dense_285 (Dense)	(None, 13)	13325

Total params: 58,317
Trainable params: 58,317
Non-trainable params: 0

La primera capa convolucional posee 128 filtros y la segunda un total de 64, aunque ambas con un kernel de 5. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y una densa, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **90.152%** (+/-0.553) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 70.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_73 (Conv1D)	(None, 33, 256)	12544
dropout_43 (Dropout)	(None, 33, 256)	0
max_pooling1d_43 (MaxPoolin g1D)	(None, 16, 256)	0
flatten_130 (Flatten)	(None, 4096)	0
dense_307 (Dense)	(None, 1000)	4097000
dense_308 (Dense)	(None, 13)	13013

Total params: 4,122,557
Trainable params: 4,122,557
Non-trainable params: 0

La capa convolucional posee 256 filtros con un kernel de 8. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y una

función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **90.470%** (+/-1.472) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 70.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

Layer (type)	Output Shape	Param #
conv1d_24 (Conv1D)	(None, 36, 128)	3968
conv1d_25 (Conv1D)	(None, 32, 64)	41024
dropout_12 (Dropout)	(None, 32, 64)	0
max_pooling1d_12 (MaxPoolin g1D)	(None, 16, 64)	0
flatten_99 (Flatten)	(None, 1024)	0
dense_265 (Dense)	(None, 1000)	1025000
dense_266 (Dense)	(None, 13)	13013

Total params: 1,083,005
Trainable params: 1,083,005
Non-trainable params: 0

La primera capa convolucional posee 128 filtros y la segunda un total de 64, aunque ambas con un kernel de 8. Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y una función de activación ReLU, y la segunda, con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **90.061%** (+/-0.972) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 70.

5.4.2.2.2. Resumen redes convolucionales 1D

En la Tabla 23, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 23. Resumen resultados redes convolucionales 1D dominio temporal excluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	88.103	90.152	90.470	90.061
Accuracy máximo (%)	88.802	90.705	91.942	91.033

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 60:

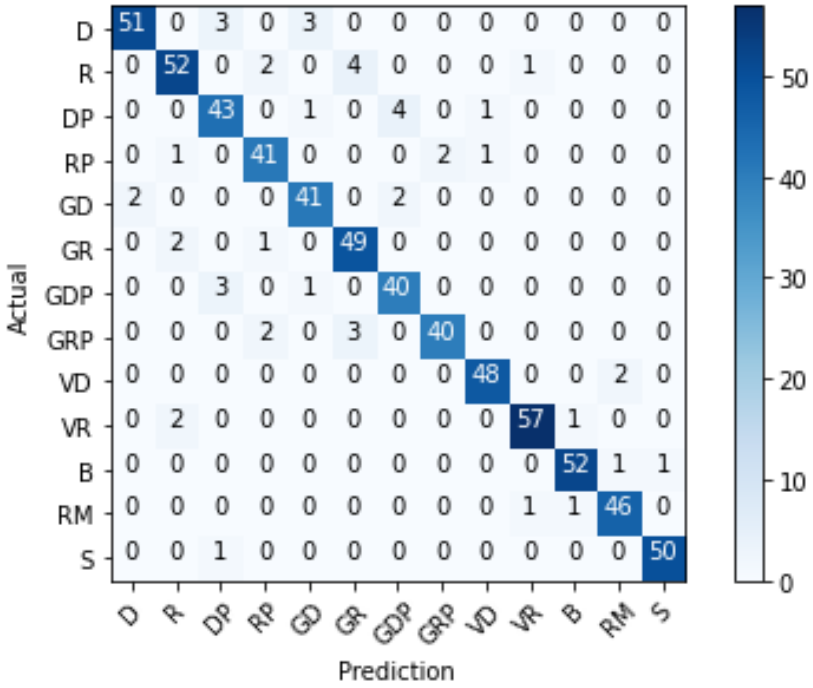


Figura 60. Matriz confusión mejor configuración redes convolucionales 1D en el dominio temporal excluyendo deportistas zurdos

Los números más altos fuera de la diagonal principal, es decir, los golpes que más le cuesta distinguir al algoritmo han sido: confundir un revés con un globo de revés y un globo de derecha con pared con un globo de derecha, acumulando un total de 4 predicciones erróneas en cada uno.

5.4.2.3. Redes convolucionales 2D

5.4.2.3.1. Una capa convolucional y una capa densa

La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_94 (Conv2D)           (None, 38, 4, 128)        1280
dropout_60 (Dropout)         (None, 38, 4, 128)        0
max_pooling2d_67 (MaxPoolin  (None, 19, 2, 128)        0
g2D
flatten_58 (Flatten)         (None, 4864)              0
dense_83 (Dense)             (None, 13)                63245
=================================================================
Total params: 64,525
Trainable params: 64,525
Non-trainable params: 0
```

Posee una primera capa convolucional con 256 filtros y un kernel de (3,3). Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **82.914%** (+/-

1.034) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y una capa densa La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_74 (Conv2D)           (None, 38, 4, 128)        1280
conv2d_75 (Conv2D)           (None, 36, 2, 256)        295168
dropout_48 (Dropout)         (None, 36, 2, 256)        0
max_pooling2d_55 (MaxPoolin  (None, 18, 1, 256)        0
g2D)
flatten_46 (Flatten)         (None, 4608)              0
dense_71 (Dense)             (None, 13)                59917
=================================================================
Total params: 356,365
Trainable params: 356,365
Non-trainable params: 0
```

Posee dos capas convolucionales, la primera con 128 filtros y la segunda con 256, ambas con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*, cuya finalidad es extraer las características más importantes recogidas por la capa convolucional. A continuación, se añade una capa *flatten* como predecesora a la densa. Esta última posee una cantidad de 13 neuronas con función de activación *softmax*. El mejor accuracy logrado es **87.557%** (+/-1.191) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

Una capa convolucional y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_64 (Conv2D)           (None, 36, 2, 128)        3328
dropout_42 (Dropout)         (None, 36, 2, 128)        0
max_pooling2d_49 (MaxPoolin  (None, 18, 1, 128)        0
g2D)
flatten_40 (Flatten)         (None, 2304)              0
dense_61 (Dense)             (None, 1000)              2305000
dense_62 (Dense)             (None, 13)                13013
=================================================================
Total params: 2,321,341
Trainable params: 2,321,341
```

La capa convolucional posee 256 filtros con un kernel de (3,3). Seguidamente se añade una capa *dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas, la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **89.514%** (+/-1.464) y se ha dado para 70 *epochs* y un tamaño de *batch size* de 30.

Dos capas convolucionales y dos capas densas La configuración del clasificador que tuvo una mayor accuracy fue la siguiente:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_40 (Conv2D)           (None, 38, 4, 128)        1280
conv2d_41 (Conv2D)           (None, 36, 2, 256)        295168
dropout_27 (Dropout)         (None, 36, 2, 256)        0
max_pooling2d_34 (MaxPoolin  (None, 18, 1, 256)        0
g2D)
flatten_25 (Flatten)         (None, 4608)              0
dense_31 (Dense)             (None, 1000)              4609000
dense_32 (Dense)             (None, 13)                13013
=================================================================
Total params: 4,918,461
Trainable params: 4,918,461
Non-trainable params: 0
```

La primera capa convolucional posee 128 filtros y la segunda 256, ambas con un kernel de 5. Seguidamente se añade una capa *dropout dropout* y otra *maxpooling*. A continuación, se añaden una capa *flatten* y dos densas: la primera con 1000 neuronas y función de activación ReLU y la segunda con 13 neuronas y función de activación *softmax*. El mejor accuracy logrado es **90.334%** (+/-1.624) y se ha dado para 150 *epochs* y un tamaño de *batch size* de 30.

5.4.2.3.2. Resumen redes convolucionales 2D

En la Tabla 7, se muestra un resumen de los distintos accuracy obtenidos para las distintas configuraciones probadas, donde "C" representa una capa Convolucional y "D" una capa Densa.

Tabla 24. Resumen resultados redes convolucionales 2D dominio temporal excluyendo deportistas zurdos

	1C y 1D	2C y 1D	1C y 2D	2C y 2D
Accuracy medio (%)	82.914	87.557	89.514	90.334
Accuracy máximo (%)	83.948	88.748	90.978	91.958

Además, la matriz de confusión para el caso de la mejor configuración se puede ver en la Figura 61:

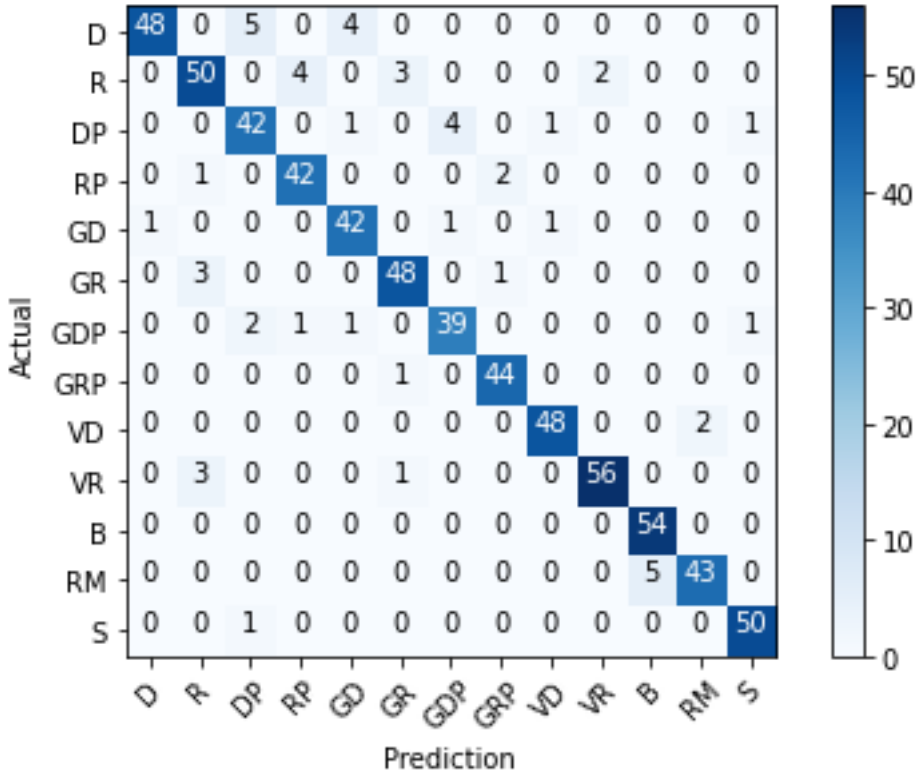


Figura 61. Matriz confusión mejor configuración redes convolucionales 2D en el dominio temporal excluyendo deportistas zurdos

Los números más altos fuera de la diagonal principal, es decir, los golpes que más le cuesta distinguir al algoritmo han sido: confundir un remate con una bandeja y una derecha con una derecha con pared, acumulando un total de 5 predicciones erróneas en cada uno.

5.4.2.4. Árbol de decisión

Para encontrar los valores de los hiperparámetros con los que el árbol de decisión obtiene una mayor cantidad de aciertos se realiza la misma búsqueda de rejilla que en el apartado anterior, con los valores mostrados en la Tabla 8.

La mayor accuracy obtenida es del **54.874%** (+/-0.636) y se ha dado para una *max_depth* de 30, un *min_samples_split* de 2, un *min_samples_*leaf de 1 y un *criterion* de entropy. La matriz de confusión para este caso se puede observar en la siguiente Figura 62:

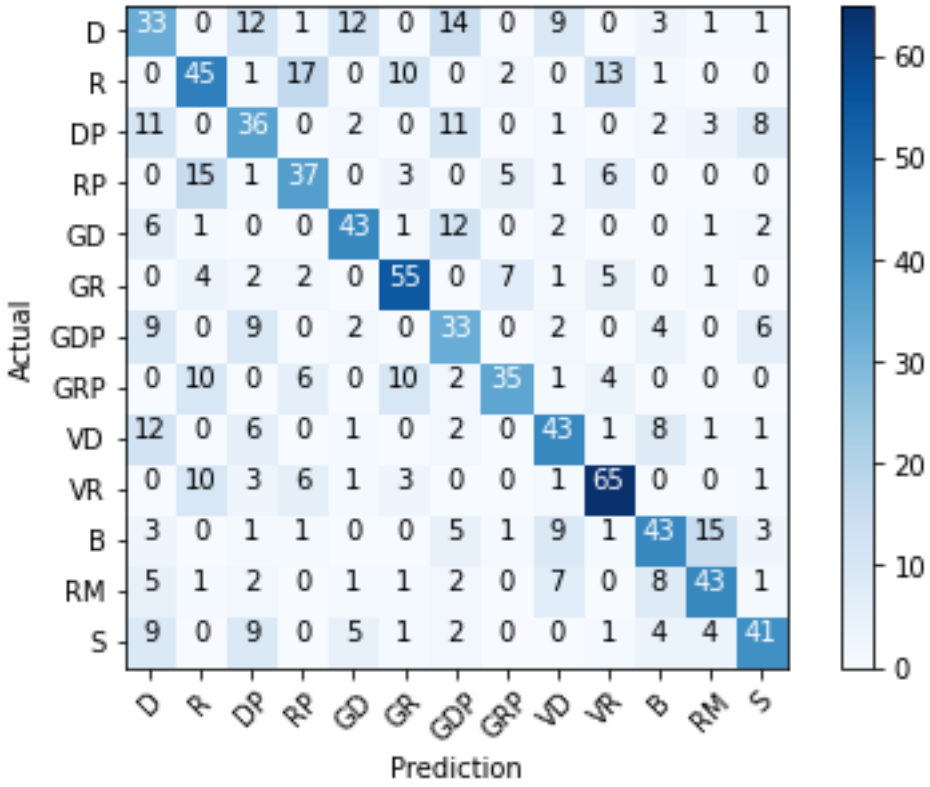


Figura 62. Matriz confusión mejor configuración árbol de decisión en el dominio temporal excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el reves con pared con el revés, ya que los ha clasificado mal hasta 17 veces. Además, presenta dificultades para diferenciar el remate de la bandeja y el reves con pared con el revés, ya que la técnica de ejecución es bastante parecida. Movimientos como el remate con el globo de revés con pared, no tienen nada que ver es por eso que el algoritmo no presenta problemas para su diferenciación.

5.4.2.5. SVM

Los hiperparámetros que se han modificado con el fin de obtener una mejor accuracy son los mismos que en el apartado anterior: la regularización (C) y el filtro kernel.

Tabla 25. Resultados SVM en dominio temporal excluyendo deportistas zurdos, en función de kernel y C

		Regularización ©								
		0.01	0.1	0.5	1	2	10	12	20	100
Filtro kernel	Lineal	70.58	65.62	63.70	63.90	63.90	63.90	63.90	63.90	63.90
	Polinómico	18.50	48.43	74.22	81.40	84.73	87.16	87.36	87.46	87.56
	Radial	14.66	52.88	78.56	85.04	89.08	90.90	90.70	90.70	90.60
	Sigmoide	15.67	31.95	36.50	32.66	28.72	22.65	22.65	22.35	20.93

La mayor accuracy se ha dado para el filtro kernel de tipo radial y una C de 10, obteniendo un **90.90%** de aciertos. La matriz de confusión para este caso se puede observar en la siguiente Figura 63:

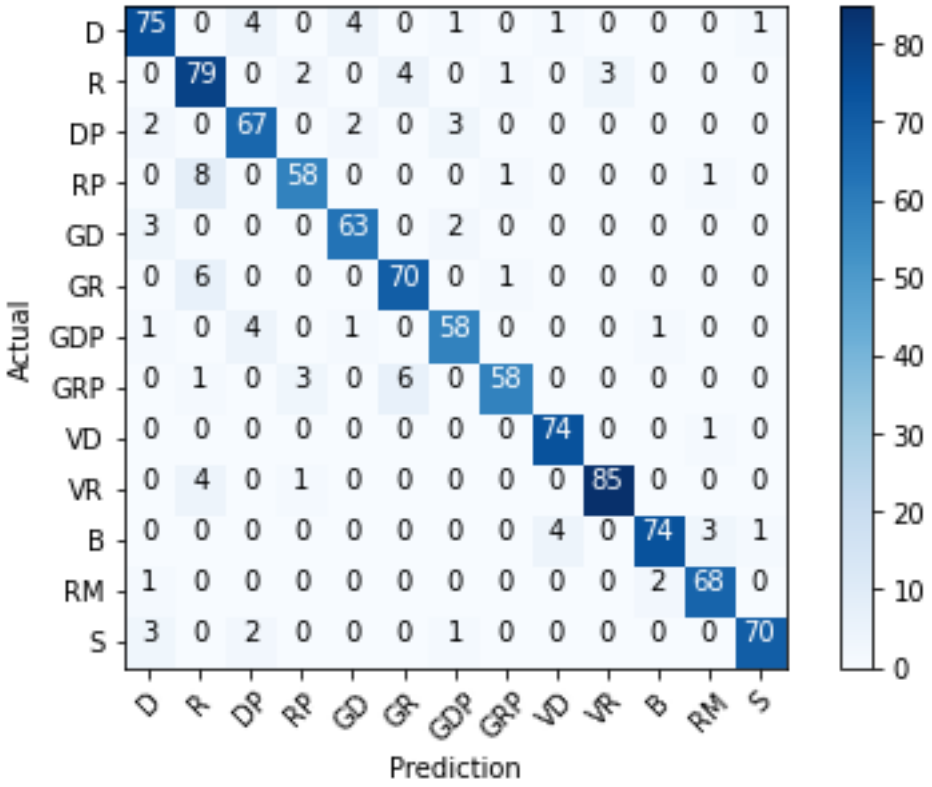


Figura 63. Matriz confusión mejor configuración en SVM en el dominio temporal excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido el revés con pared con el revés, ya que los ha clasificado mal hasta 8 veces.

5.4.2.6. KNN

valor de k, es decir, la cantidad de vecinos más próximos que se tienen en cuenta para la clasificación. Se ha probado para distintos valores de k, desde el 1 (tomando tan solo el vecino más cercano) hasta 30

Tabla 26. Resultados KNN en dominio temporal excluyendo deportistas zurdos, en función de k

k	1	2	3	4	5	6	7	8	9
Accuracy (%)	83.62	77.05	79.17	77.55	77.86	76.54	76.95	75.83	76.04
k	10	11	12	13	14	15	20	25	30
Accuracy (%)	74.52	73.61	74.12	73.10	72.30	71.99	68.66	65.72	64.71

El mejor resultado se da para k = 1 con un accuracy del **83.62 %**. La matriz de confusión para este caso se puede observar en la siguiente Figura 64:

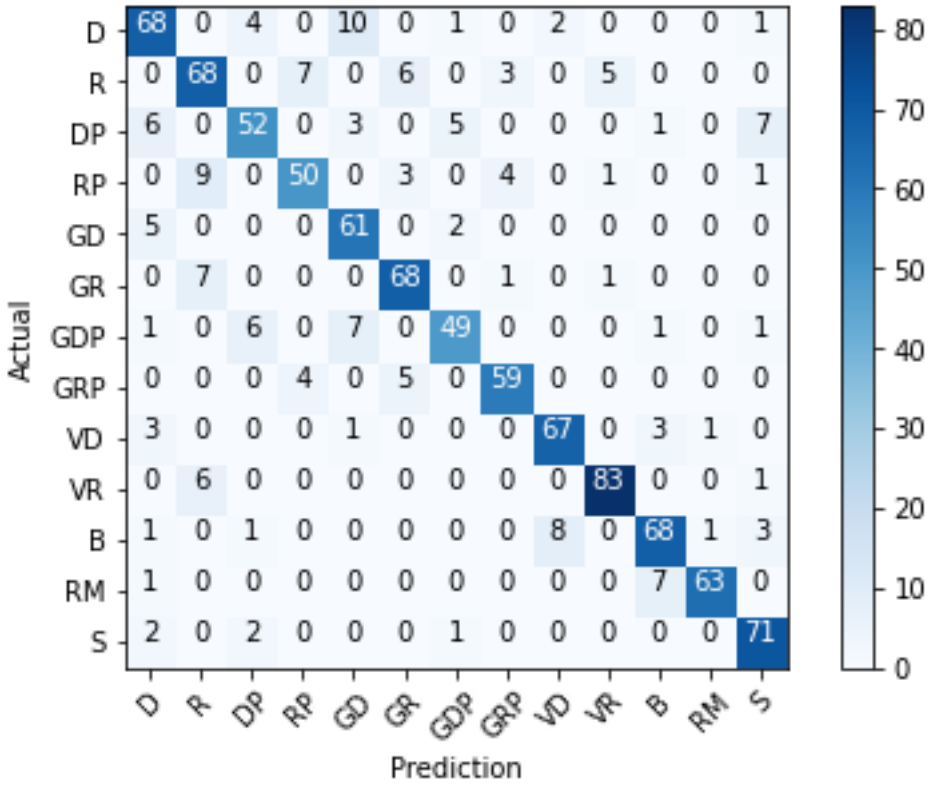


Figura 64. Matriz confusión mejor configuración en KNN en el dominio temporal excluyendo deportistas zurdos

El número más alto fuera de la diagonal principal, es decir, los golpes que más confunde el algoritmo han sido la derecha con el globo de derecha, ya que los ha clasificado mal hasta 10 veces.

5.4.3. Resumen de mejores resultados en el dominio del tiempo

En la Tabla 27, se muestra un resumen de los resultados de los experimentos. En ella se puede observar el mejor accuracy de cada algoritmo en el dominio temporal, para los casos en los que se han incluido golpes de deportistas zurdos y en los que no.

Tabla 27. Resumen resultados en el dominio temporal.

		Con zurdos		Sin zurdos	
		Accuracy medio (%)	**Accuracy máximo (%)**	**Accuracy medio (%)**	**Accuracy máximo (%)**
Redes neuronales densas	Dos capas densas	89.075	89.462	90.212	90.734
	Tres capas densas	89.137	89.559	91.062	91.529
Redes neuronales convolucionales 1D	Una capa convolucional y una capa densa	84.303	85.282	88.103	88.802
	Dos capas convolucionales y una capa densa	87.904	88.578	90.152	90.705

	Una capa convolucional y dos capas densas	87.633	90.301	90.470	91.942
	Dos capas convolucionales y dos capas densas	88.471	89.073	90.061	91.033
Redes neuronales convolucionales 2D	Una capa convolucional y una capa densa	81.578	82.268	82.914	83.948
	Dos capas convolucionales y una capa densa	85.656	87.605	87.557	88.748
	Una capa convolucional y dos capas densas	86.621	88.090	89.514	90.978
	Dos capas convolucionales y dos capas densas	88.101	90.049	90.334	91.958
Árbol de decisión		53.898	54.771	54.874	55.510
SVM		90.540	91.278	90.900	91.322
KNN		81.830	83.211	83.620	85.021

Como conclusión general del dominio temporal, el mejor accuracy medio para el caso en el que se entrena el modelo con datos tanto de diestros como de zurdos, se consigue con el algoritmo de las máquinas vector soporte alcanzando un 90.54% de aciertos de media llegando en una ocasión al 91.278% de clasificaciones correctas. En cambio, en el caso de entrenar los algoritmos con datos exclusivamente de diestros el accuracy medio más alto es del 91.062% y se alcanza para las redes densa con tres capas, aunque se logra llegar al 91.958% de clasificaciones correctas en las redes convolucionales 2D, con dos capas densas y dos convolucionales 2D.

En líneas generales, el hecho de introducir jugadores zurdos en la base de datos influye negativamente en el dominio del tiempo. A pesar de entrenar los algoritmos con una mayor cantidad de datos, se obtienen peores resultados, llegando a obtener un 2.837% menos de aciertos de media en el caso de las redes convolucionales 1D con una capa convolucional y dos desas Esto demuestra que los deportistas zurdos realizan movimientos distintos en ciertos golpeos, por lo que si se utilizan sus datos para entrenar al modelo que clasificará golpes de diestros, se hará de forma errónea.

5.5. Comparación dominio temporal y frecuencial

En la Tabla 28 se muestra un resumen de resultados de la accuracy obtenida para todos los entrenamientos, tanto en el dominio temporal como en el frecuencial, donde “D” representa una capa densa y “C” una capa convolucional. En la columna “máximo”, se representa el accuracy máximo que se ha obtenido entre todos los experimentos de ese mismo algoritmo. Además, en las columnas “TE” y “TP” se muestra el Tiempo de Entrenamiento del algoritmo y Tiempo de Predicción, es decir, el tiempo que tarda en predecir el algoritmo la clase de la muestra. Tanto el TE como el TP se han obtenido para el entrenamiento que ha obtenido el mayor accuracy.

Tabla 28. Resumen resultados globales.

		DOMINIO FRECUENCIAL		DOMINIO TEMPORAL				
		Con zurdos	**Sin zurdos**	**Con zurdos**	**Sin zurdos**	**Máximo**	**TE (s)**	**TP (s)**
Redes neuronales densas	2D	73.946	72.473%	89.07	90.212	90.734	194.15	0.207
	3D	79.864	78.498	89.137	91.062	91.529		
Redes neuronales convolucionales 1D	1C y 1D	70.801	71.821	84.303	88.103	88.802	690.10	0.358
	2D y 1D	72.972	72.489	87.904	90.152	90.705		
	1C y 2D	68.015	67.769	87.633	90.470	91.942		
	2C y 2D	70.197	69.181	88.471	90.061	91.033		
Redes neuronales convolucionales 2D	1C y 1D	58.940	66.950	81.578	82.914	83.948	1250.18	0.933
	2D y 1D	75.216	77.329	85.656	87.557	88.748		
	1C y 2D	74.057	74.962	86.621	89.514	90.978		
	2C y 2D	75.413	76.024	88.101	90.334	91.958		
Árbol de decisión		44.762	47.128	53.898	54.874	55.510	20.85	0.254
SVM		80.760	82.100	90.540	90.900	91.322	0.197	0.704
KNN		79.110	80.590	81.830	83.620	85.021	0.0063	0.0784

El entrenamiento que maximiza el accuracy se da para las redes neuronales con tres capas densamente conectadas, entrenando el modelo con la base de datos en el dominio temporal con jugadores exclusivamente diestros. Logra obtener un porcentaje de aciertos del 91.062% de media, un resultado bastante aceptable para un clasificador de 13 clases. En el caso de realizar el entrenamiento con jugadores tanto diestros como zurdos, la SVM logra el máximo en el dominio temporal con un 90.54% de clasificaciones acertadas. El porcentaje máximo absoluto de clasificaciones correctas lo obtiene el clasificador con una capa convolucional y dos capas densas, habiendo sido entrenado con la base de datos en el dominio temporal y excluyendo a jugadores zurdos de esta. Logra alcanzar un 91.942% de aciertos.

Con respecto al tiempo que tarda cada uno en clasificar la muestra de entrada, se puede observar que ninguno de ellos tarda más de un segundo en realizar la clasificación. Esto implica la posibilidad de utilizar los modelos

entrenados durante un partido, ya que físicamente no es posible realizar dos golpes distintos en menos de un segundo.

A rasgos generales, se puede observar que con la base de datos en el dominio temporal se obtienen resultados bastante superiores, llegando, en ocasiones, a superar en un 20% el porcentaje de clasificaciones correctas. Por ejemplo, en las redes convolucionales 1D con dos capas densas y dos capas convolucionales, el entrenamiento en el dominio del tiempo sin zurdos, supera por un 20.88% al de la frecuencia.

En las redes neuronales densas, tanto en el dominio frecuencial como en el temporal, al aumentar el número de capas se aumenta el porcentaje de aciertos, aunque no de manera significativa en el dominio temporal aumentando apenas un 1% el accuracy. El dominio de la frecuencia sé se ve más afectado cuando se incrementan las capas, llegando a aumentar hasta un 6.025% el accuracy. En el dominio frecuencial, la adición de datos de jugadores zurdos es indiferente, importando únicamente la cantidad de datos para mejorar el accuracy, es decir, a mayor cantidad de datos, mayor cantidad de aciertos. En cambio, en el domino temporal, para mejorar la clasificación se ha de entrenar la red con datos de jugadores del mismo tipo que se quieran clasificar.

Para el caso de las redes convolucionales 1D, resulta lógico que el dominio del tiempo obtenga mejores resultados que el frecuencial, ya que es un clasificador diseñado concretamente para recibir series temporales como entrada. Tanto en el dominio del tiempo como en el frecuencial, la adición de capas convolucionales 1D provoca una mejora en los resultados de las clasificaciones. En el caso de la frecuencia, añadir capas densas hace empeorar los resultados, al contrario que en el dominio del tiempo que cualquier aumento de capas mejoran los resultados. Al igual que en el caso de las redes densas, el aumento o disminución de capas provoca cambios más significativos en el dominio frecuencial.

A diferencia de las redes convolucionales 1D, los algoritmos con capas convolucionales 2D entrenadas con el dominio frecuencial, el hecho de tratar los datos como imágenes de 6x20 (GDL x nº de muestras) hace incrementar el accuracy medio. El mayor accuracy medio se da para el caso de dos capas convolucionales 2D y dos densas, y se logra un 76.024% de clasificaciones correctas, frente al 66.950% en el caso de una capa convolucional 2D y una densa. En el dominio del tiempo, también influye positivamente en el resultado la adición de capas convolucionales 2D, alcanzando el máximo global en una ocasión donde se obtiene hasta un 91.958% de clasificaciones correctas.

Lo último a destacar es que el clasificador considerado más simple, el KNN con k = 1, es decir, asignar la clase del vecino más cercano, obtiene un porcentaje de aciertos muy similar al mejor, distando de este en tan solo un 0.162%.

Para finalizar, cabe destacar que desde un principio se cuenta con la clasificación errónea de aquellos golpes que presentan una técnica muy parecida. Por ejemplo, resulta muy sencillo diferenciar un golpe de revés con pared con una volea de derecha, pero una bandeja realizada con el brazo un poco más estirado de lo normal se puede convertir en un remate. Lo que hace la distinción de ambos muy compleja incluso a ojos de expertos en el deporte.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. Conclusiones

El entrenamiento de los modelos con la base de datos en el dominio temporal logra hasta en tres clasificadores un porcentaje de aciertos de más del 90% y en el dominio frecuencial dos de ellos superan el 80%, concluyendo los experimentos exitosamente. Se demuestra el gran potencial de los algoritmos de *machine learning* para la clasificación en deportes de raqueta.

Los algoritmos presentan un mayor grado de dificultad para realizar la clasificación de los golpes de revés, en el caso de añadir jugadores zurdos en el conjunto de datos los golpes más complicados de distinguir son, por un lado, el revés con la volea de revés y el revés con el globo de revés por el otro. En el caso de los jugadores diestros es el revés con el revés con pared. Las clasificaciones incorrectas ya habían sido previstas, puesto que las técnicas de ejecución de los golpes mal clasificados resultan muy similares, en ocasiones no diferenciables por el ojo humano.

Los resultados en frecuencia, a pesar de ser de más bajo rendimiento que los temporales, continúan siendo bastante buenos, ya que, se debía entrenar un clasificador con una base de datos de tan solo 4002 golpes para representar a 13 clases distintas. Es probable que la inferioridad de estos resultados haya sido causada por la baja frecuencia de muestreo (20 Hz), suponiendo así una pérdida de información. Si se aumentara la frecuencia de muestreo, se podría mejorar el accuracy de los algoritmos entrenados con el dominio frecuencial.

Una vez escogido el dominio temporal como mejor para realizar la clasificación, se observa la importancia de introducir jugadores zurdos en el conjunto de datos. Se corrobora la presuposición inicial de que estos presentan distintas señales recogidas por la IMU en ciertos golpeos. Esto conlleva a confundir al clasificador haciendo que funcione por debajo de sus posibilidades.

La conclusión del estudio se puede resumir en la elección del dominio temporal para la clasificación de golpes de padel mediante algoritmos de aprendizaje automático. Concretamente, un clasificador de redes neuronales densamente conectadas con 3 capas, ya que además del elevado porcentaje de aciertos, los tiempos tanto de entrenamiento como de predicción, son bastante reducidos si se comparan con el resto de algoritmos que también presentan un accuracy medio alto. Aunque si se desea tener un tiempo de entrenamiento especialmente rápido a costa de un ínfimo porcentaje de clasificaciones correctas inferior, se debe escoger el algoritmo SVM, que también presenta un accuracy medio bastante elevado. Además, la adición de jugadores zurdos al conjunto complica la clasificación por lo que se necesitan más datos de este tipo de deportistas para obtener unos resultados similares. No se tienen los suficientes datos de jugadores zurdos para llegar a la conclusión de la necesidad de crear dos bases de datos independientes para entrenar los algoritmos.

6.2. Trabajos futuros

Los trabajos futuros relacionados con este proyecto se pueden dividir en dos líneas de trabajo. Por un lado, la mejora del conjunto de datos y por otro el perfeccionamiento de los algoritmos de aprendizaje automático. La primera línea resulta de especial interés, ya que no existen bases de datos que recojan golpes de pádel.

En este estudio tan solo se han considerado 13 golpes a clasificar de todos los posibles que existen. Una primera mejora sería incluir nuevos golpes, como la chiquita, la víbora, la sacada por tres y por cuatro, la salida de pared lateral, jugadas con la doble pared, etc.

Además, los golpes que se han realizado han sido en su totalidad planos. Otro punto a perfeccionar es la inclusión de efectos en los distintos golpeos, es decir, el clasificador no solo diferenciará el golpe de entrada, si no que será capaz de visualizar si lleva efecto y el tipo de este. Además, los efectos han podido ser una fuente de errores en la clasificación por lo que se podría mejorar el porcentaje de aciertos del presente estudio.

Otra fuente de error puede ser el hecho de tener una única base de datos para todos los niveles de juego. Hay golpes que en niveles bajos no se enseñan, así como los efectos. Además, los movimientos en niveles

profesionales distan mucho de los realizados en niveles inferiores por lo que, puede dar lugar a una identificación errónea del golpe.

Se propone realizar ampliar la base de datos, tomando muestras de nuevos jugadores, tanto diestros como zurdos, tanto con efectos como con golpes nuevos. También, se propone la realización de diferentes bases de datos en función del nivel del jugador, con el fin de observar los resultados en el accuracy de los algoritmos. Asimismo, se sugiere la diferenciación entre mujeres y hombres, ya que estos últimos, en la mayoría de los casos, presentan un juego más agresivo, pudiendo verse esto afectado en los datos de las aceleraciones y velocidades.

Además, las bases de datos utilizadas en el entrenamiento de los algoritmos de aprendizaje automático utilizan miles de datos, aunque la cantidad de estos depende tanto del algoritmo utilizado como del problema a estudiar. Cuantos más datos se utilicen para el entrenamiento, mejores resultados se obtendrán. En este caso la obtención de los datos resulta costosa ya que, se necesita la disponibilidad de una pista de padel, del material necesario, de deportistas dispuestos a prestar su tiempo para la realización de las pruebas, etc.

Una solución al problema anterior puede ser la técnica conocida como *data augmentation*, que consiste en producir alteraciones en los datos originales ya recogidos. De esta forma se permite aumentar tanto en variedad como en cantidad el conjunto de datos antiguo, controlando el *overfitting* [64]. Un ejemplo de esto se puede apreciar en la Figura 65, donde para entrenar un algoritmo de CNN en *computer visión* se transforma la foto de un gato. De la foto original se obtienen 6 nuevas.

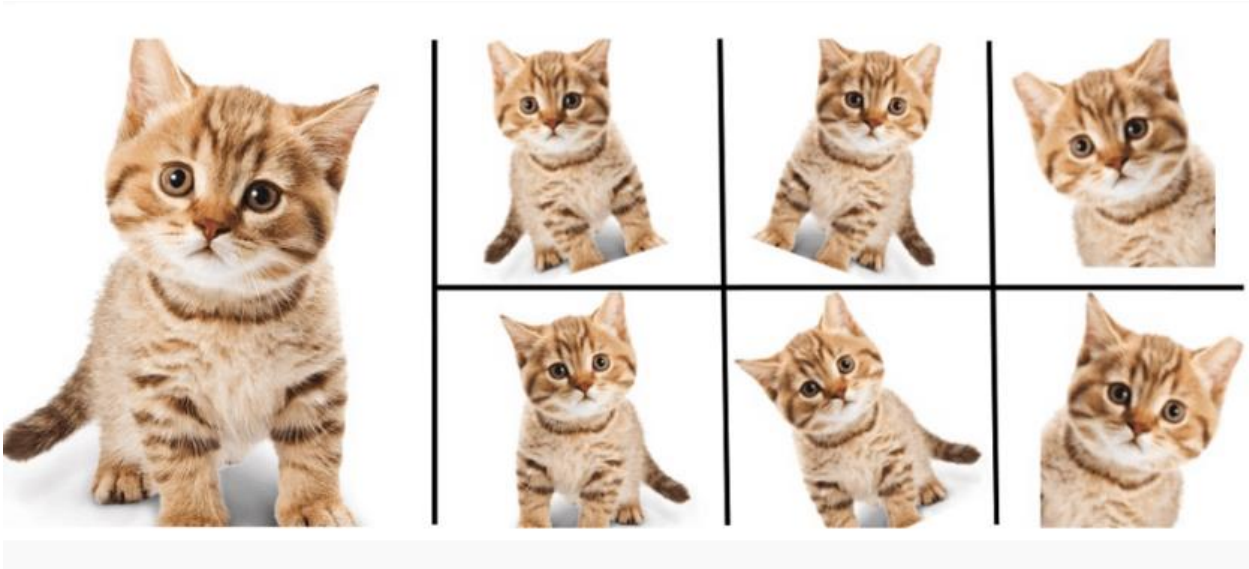


Figura 65. Ejemplo de *data augmentation*. Disponible en [64].

Otro aspecto a mejorar es el mecanismo de toma de datos, ya que el aparato utilizado para recoger los datos resulta aparatoso para realizar los movimientos. Lo ideal sería diseñar un dispositivo con lo indispensable para tomar las muestras, es decir, con la IMU, de tal forma que este interfiera lo menos posible en las jugadas del deportista.

REFERENCIAS

[1] «PERIÓDICO UNIVERSITARIO UCM EN LA WEB 2.0,» [En línea]. Available: https://generaciondospuntocero.com/boom-fenomeno-fitness/#:~:text=El%20fen%C3%B3meno%20fitness%2C%20se%20puede,al%20propio%20ejercicio%20que%20realizamos.. [Último acceso: 15 05 2022].

[2] «Dispositivos Wearables,» [En línea]. Available: https://www.dispositivoswearables.net/. [Último acceso: 15 05 2022].

[3] R. Fernández, «Statista,» 23 09 2021. [En línea]. Available: https://es.statista.com/estadisticas/702760/envios-mundiales-de-wearables-en-unidades/. [Último acceso: 15 05 2022].

[4] «Diario de Sevilla,» [En línea]. Available: https://www.diariodesevilla.es/tecnologia/mercado-wearables-2021-apple_0_1664533775.html. [Último acceso: 15 05 2022].

[5] «Threepoints, the school for digital business,» 26 05 2020. [En línea]. Available: https://www.threepoints.com/blog/big-data-aplicado-a-los-deportes. [Último acceso: 15 05 2022].

[6] P. G. Bejerano, «ThinkBig,» [En línea]. Available:https://blogthinkbig.com/big-data-y-deporte-cuando-el-partido-se-juega-con-algoritmos#:~:text=%E2%80%9CEl%20uso%20del%20big%20data,valor%20a%C3%B1adido%20a%20tu%20disposici%C3%B3n%E2%80%9D.. [Último acceso: 15 05 2022].

[7] G. C. Domínguez, Comparación de algoritmos de aprendizaje automático para clasificación de golpes de pádel, Sevilla, 2021.

[8] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Dominio_de_la_frecuencia. [Último acceso: 16 05 2022].

[9] J. N. T. S. y. J. C. Joseph McGrath, «Upper body activity classification,» *Journal of Sports Engineering an Technology,* 16 07 2019.

[10] M. Z. U. A. M. y. A. A. Mohammed Mehedi Hassan, «A robust human activity recognition system using smartphone,» *ELSEVIER,* 31 07 2017.

[11] R. S.-S. F. F.-M. J. F.-L. Manuel Gil-Martín, «Improving physical activity recognition using a new deep learning architecture and post-processing techniques,» *ELSEVIER,* 27 03 2019.

[12] M. F. Y. H. R. W. Y. W. Y. L. S. W. G. X. Mingyue Wu, «A real-time tennis level evaluation and strokes classification system based on the Internet of Things,» *ELSEVIER,* 19 09 2021.

[13] C. M. Fernández, «Universidad Politécnica de Madrid,» Enero 2020. [En línea]. Available: https://oa.upm.es/62682/1/TESIS_MASTER_CLARA_MENDUINA_FERNANDEZ.pdf. [Último acceso: 02 07 2022].

[14] B. J. S.-A. J. G.-R. S. J. I. Adrián Escudero-Tena, «Analysis of Game Performance Indicators during 2015–2019 World Padel Tour Seasons and Their Influence on Match Outcome,» *Environmental Research and Public Health,* 29 03 2021.

[15] A. B. Polo, «New Padel Blog,» 17 07 2017. [En línea]. Available: https://www.newpadel.net/blog/cuales-las-medidas-una-pista-padel/#:~:text=Las%20medidas%20de%20una%20pista%20de%20p%C3%A1del%20oficial%20deben%20ser,de%2010%C3%9710%20metros.&text=Entre%20la%20pared%20y%20la,hay%203%20metros%20de%20distancia.. [Último acceso: 25 05 2022].

[16] «Padel Galis,» [En línea]. Available: https://www.padelgalis.com/tipo-de-pistas/. [Último acceso: 25 05 2022].

[17] «Zona de Padel,» [En línea]. Available: https://www.zonadepadel.es/blog/2018/09/reglas-del-padel-reglamento-oficial/#:~:text=La%20pala%20no%20podr%C3%A1%20medir,sacar%20dentro%20del%20cuadrado%20opuesto.. [Último acceso: 2022 05 25].

[18] «Padel Addict,» [En línea]. Available: https://www.padeladdict.com/coleccion-de-padel-wilson-2019/. [Último acceso: 2022 05 25].

[19] R. Solé, «Profesional Review,» 18 07 2021. [En línea]. Available: https://www.profesionalreview.com/2021/07/18/que-es-raspberry-pi/. [Último acceso: 25 05 2022].

[20] «El Blog de Chito de Tecnología,» 04 01 2017. [En línea]. Available: https://chitoraspberrypi.blogspot.com/2017/01/sense-hat.html. [Último acceso: 2022 05 25].

[21] «Datademia,» [En línea]. Available: https://datademia.es/blog/que-es-python. [Último acceso: 2022 05 25].

[22] «Oficina de Seguridad del Internauta,» [En línea]. Available: https://www.osi.es/es/herramientas-gratuitas/vnc-viewer. [Último acceso: 2022 05 25].

[23] «Sawakinome,» [En línea]. Available: https://es.sawakinome.com/articles/physics-science-nature/difference-between-time-domain-and-frequency-domain.html. [Último acceso: 30 05 2022].

[24] A. Nieto, «Xakata,» 09 04 2021. [En línea]. Available: https://www.xataka.com/otros/alguien-ha-hecho-video-perfecto-para-todos-que-sufrimos-intentando-

entender-transformada-fourier. [Último acceso: 30 05 2022].

[25] F. Zapata, «Lifeder,» [En línea]. Available: https://www.lifeder.com/series-de-fourier/. [Último acceso: 2022 05 30].

[26] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Transformada_de_Fourier_discreta. [Último acceso: 2022 05 31].

[27] «Universidad Nacional de Mar de Plata,» [En línea]. Available: http://www3.fi.mdp.edu.ar/tds/material/5-Errores_DFT.pdf. [Último acceso: 2022 05 31].

[28] «Lidefer,» [En línea]. Available: https://www.lifeder.com/transformada-discreta-de-fourier/. [Último acceso: 2022 05 31].

[29] Á. F. García, «Universidad del Pais Vasco, ingeniería de energías renovables,» [En línea]. Available: http://www.sc.ehu.es/sbweb/fisica3/datos/fourier/fourier_1.html. [Último acceso: 2022 05 31].

[30] «NTI audio,» [En línea]. Available: https://www.nti-audio.com/es/servicio/conocimientos/transformacion-rapida-de-fourier-fft. [Último acceso: 2022 05 31].

[31] «Rufián en la Red,» 2018 02 27. [En línea]. Available: https://rufianenlared.com/fft/. [Último acceso: 2022 05 31].

[32] L. Llamas, «Luis Llamas. Ingeniería, Informática y Diseño,» 09 12 2020. [En línea]. Available: https://www.luisllamas.es/explicacion-del-teorema-de-muestreo-de-nyquist-sin-ecuaciones/. [Último acceso: 2022 05 31].

[33] «Mezcla profesional,» [En línea]. Available: https://mezclaprofesional.com/que-es-el-aliasing/. [Último acceso: 2022 05 31].

[34] «NI,» 05 03 2019. [En línea]. Available: https://www.ni.com/es-es/innovations/white-papers/06/acquiring-an-analog-signal--bandwidth--nyquist-sampling-theorem-.html. [Último acceso: 2022 05 31].

[35] «Iberdrola,» [En línea]. Available: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. [Último acceso: 2022 06 01].

[36] «Tibco,» [En línea]. Available: https://www.tibco.com/es/reference-center/what-is-supervised-learning#:~:text=El%20aprendizaje%20supervisado%20es%20una,de%20manera%20expl%C3%ADcita%20d%C3%B3nde%20buscar. [Último acceso: 2022 06 01].

[37] D. Rodríguez, «AnalyticsLane,» 16 12 2019. [En línea]. Available: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/ . [Último acceso: 04 06 2022].

[38] «hmong,» [En línea]. Available: https://hmong.es/wiki/Training_set . [Último acceso: 04 06 2022].

[39] J. Brownlee, «Machine Learning Factory,» 12 09 2019. [En línea]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/#:~:text=Overfitting%3A%20Good%20performance%20on%20the,poor%20generalization%20to%20other%20data. [Último acceso: 04 06 2022].

[40] F. S. Caparrini, «Fernando Sancho Caparrini,» 16 03 2022. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=72. [Último acceso: 2022 06 05].

[41] J. M. R. D. León. [En línea]. Available: http://pedrobeltrancanessa-biblioteca.weebly.com/uploads/1/2/4/0/12405072/red_neuronal_modelo_de_hopfield.pdf. [Último acceso: 2022 06 05].

[42] F. B. Pascual, «Universidad Complutense de Madrid,» 2016. [En línea]. Available: https://eprints.ucm.es/id/eprint/64564/1/BUENOPASCUALFERNANDO.pdf. [Último acceso: 2022 06 05].

[43] R. R. Abril, «La Máquina del Oráculo,» [En línea]. Available: https://lamaquinaoraculo.com/computacion/el-modelo-neuronal-de-mcculloch-y-pitts/. [Último acceso: 2022 06 08].

[44] «El Conspirador,» 18 10 2014. [En línea]. Available: https://www.elconspirador.com/2014/10/18/conceptos-basicos-sobre-redes-neuronales/#:~:text=Una%20red%20neuronal%20es%20el,impulsos%20nerviosos%20con%20una%20neurona. [Último acceso: 2022 06 08].

[45] L. González, «aprendeIA,» 05 10 2021. [En línea]. Available: https://aprendeia.com/que-es-el-perceptron-simple-y-multicapa/. [Último acceso: 08 06 2022].

[46] D. Calvo, «Diego Calvo,» 8 12 2018. [En línea]. Available: https://www.diegocalvo.es/perceptron/. [Último acceso: 09 06 2022].

[47] D. Calvo, «Diego Calvo,» 07 12 2018. [En línea]. Available: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/. [Último acceso: 2022 06 09].

[48] J. Torres, de *Introducción Práctica con Keras (Primera Parte)*, Barcelona, 2018, p. 208.

[49] A. S. González, «UPNA,» 27 02 2020. [En línea]. Available: https://academica-e.unavarra.es/xmlui/bitstream/handle/2454/37479/Memoria.pdf?sequence=1&isAllowed=y. [Último acceso: 10 06 2022].

[50] «Digital Guide IONOS,» 08 10 2020. [En línea]. Available: https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-keras/. [Último acceso: 10 06 2022].

[51] «ICHI.PRO,» [En línea]. Available: https://ichi.pro/es/prediccion-y-redes-convolucionales-temporales-102124506152164. [Último acceso: 11 06 2022].

[52] J. Barrios, «Health Big Data,» [En línea]. Available: https://www.juanbarrios.com/redes-neurales-convolucionales/. [Último acceso: 03 07 2022].

[53] «Geeks for geeks,» 29 07 2021. [En línea]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/. [Último acceso: 11 06 2022].

[54] S. Kumar, «HereVE'GO,» 05 07 2020. [En línea]. Available: https://www.herevego.com/redes-neuronales-de-convolucion/. [Último acceso: 03 07 2022].

[55] E. R.-L. I. M. A.-B. R. A.-E. C. Guadalupe Origel-Rivas, «Tecnológico Nacional de México,» [En línea]. Available: https://rcs.cic.ipn.mx/2020_149_8/Redes%20neuronales%20artificiales%20y%20arboles%20de%20decision%20para%20la%20clasificacion%20con%20datos%20categoricos.pdf. [Último acceso: 12 06 2022].

[56] «InteractiveChaos,» [En línea]. Available: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/arbol-de-decision. [Último acceso: 12 06 2022].

[57] J. M. Heras, «IArtificial.ner,» 28 05 2019. [En línea]. Available: https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/#:~:text=Las%20M%C3%A1quinas%20de%20Vectores%20de%20Soporte%20(Support%20Vector%20Machines)%20permiten,son%20los%20vectores%20de%20soporte.. [Último acceso: 13 06 2022].

[58] J. A. Rodrigo, «Ciencia de Datos,» Diciembre 2020. [En línea]. Available: https://www.cienciadedatos.net/documentos/py24-svm-python.html. [Último acceso: 13 06 2022].

[59] J. M. Heras, «IArtificial.ner,» 19 09 2020. [En línea]. Available: https://www.iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/. [Último acceso: 13 06 2022].

[60] F. S. Caparrini, «Fernando Sancho Caparrini,» 14 12 2020. [En línea]. Available: http://www.cs.us.es/~fsancho/?e=77. [Último acceso: 15 06 2022].

[61] R. H. Ucan, «SoldAI,» 15 06 2020. [En línea]. Available: https://medium.com/soldai/m%C3%A9todo-de-los-k-vecinos-m%C3%A1s-cercanos-f8231c28f7c7. [Último acceso: 15 06 2022].

[62] franspg, «Generación de datos artificiales (Data Augmentation),» 27 01 2020. [En línea]. Available: https://franspg.wordpress.com/2020/01/27/generacion-de-datos-artificiales-data-augmentation/. [Último acceso: 01 07 2022].

[63] D. Rodríguez, «Analyticslane,» 16 12 2019. [En línea]. Available: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/ . [Último acceso: 2022 06 04].

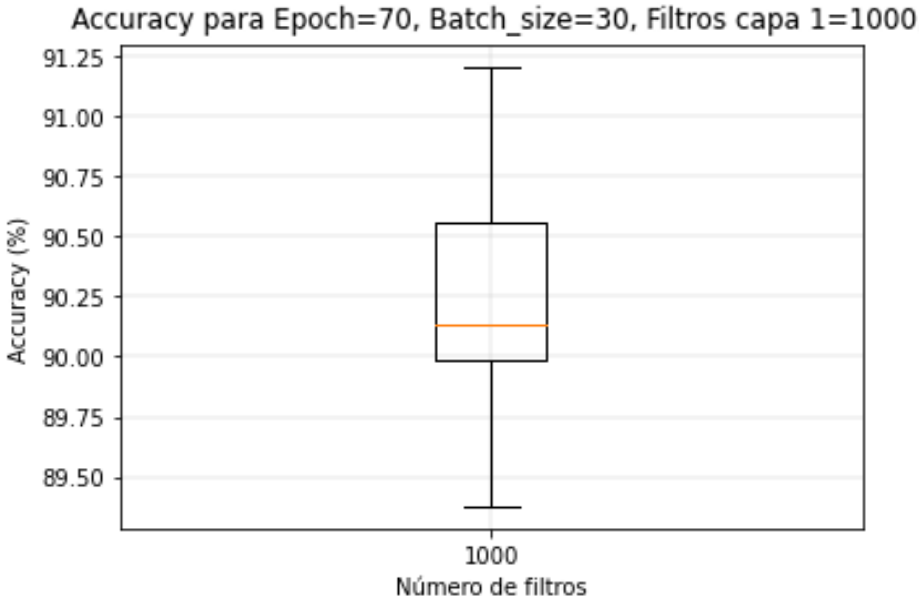
[64] R. S.-S. F. F.-M. J. F.-L. Manuel Gil-Martín, «Improving physical activity recognition using a new deep learning architecture and post-processing techniques,» *ELSEVIER,* 27 03 2019.

[65] https://github.com/cmv-81/Padel-Shot-Classification-and-Dataset-Time-and-Frecuency-Domine

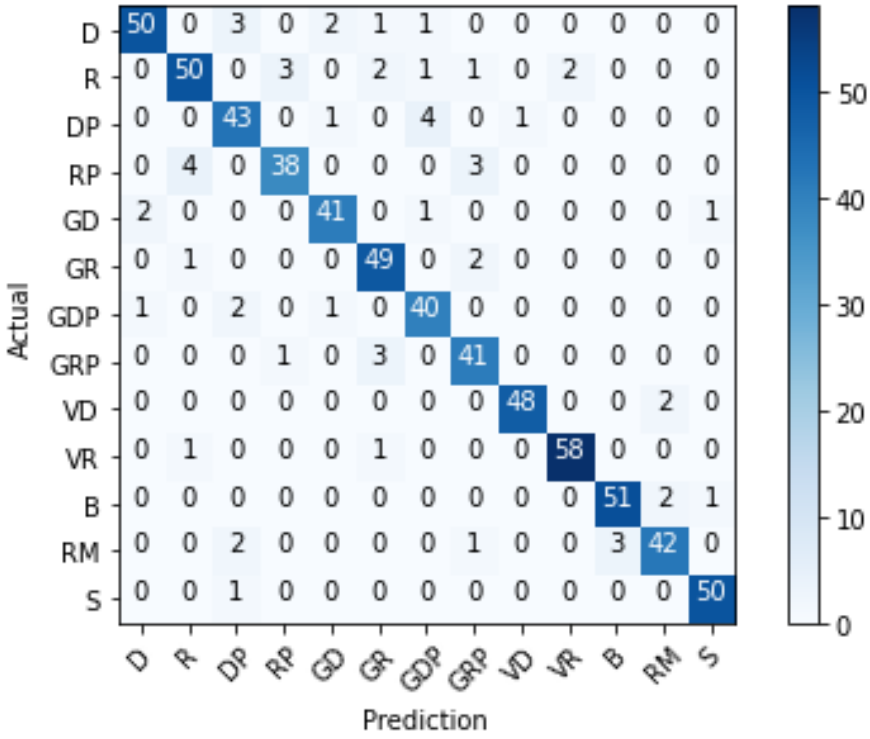
ANEXO I

A lo largo de este anexo se mostrarán los diagramas de bigotes y las matrices de confusión de las mejores configuraciones para los distintos algoritmos para el caso de estudio en el que se entrena la red neuronal con la base de datos en el dominio del tiempo y sin jugadores zurdos.

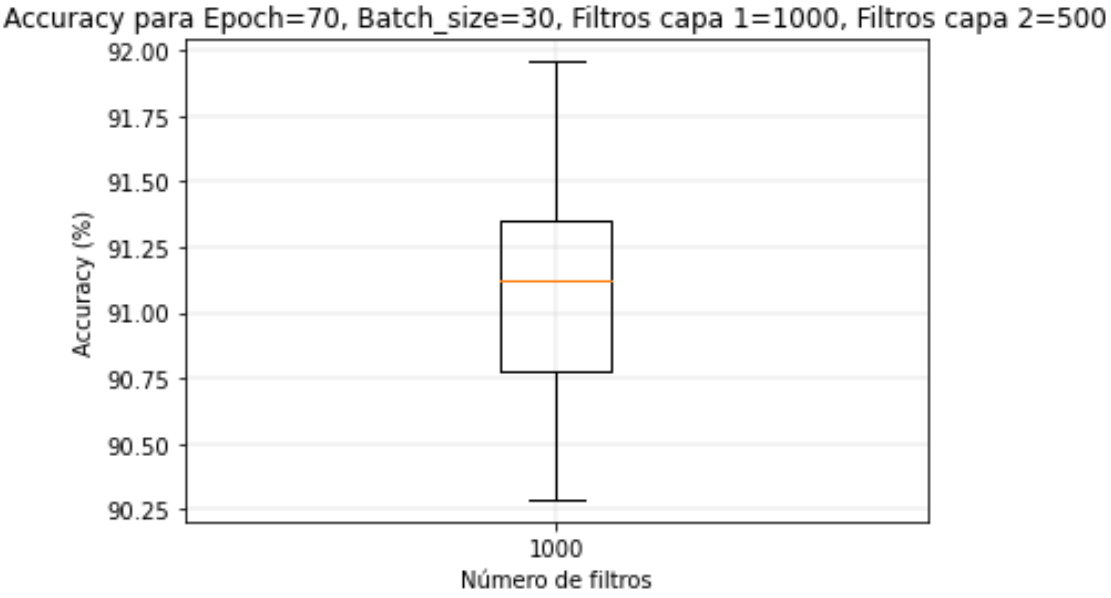
- A1.1 → REDES NEURONALES DENSAMENTE CONECTADAS
  - A1.1.1 → Dos capas densamente conectadas
    - Diagrama de bigotes



    - Matriz de confusión

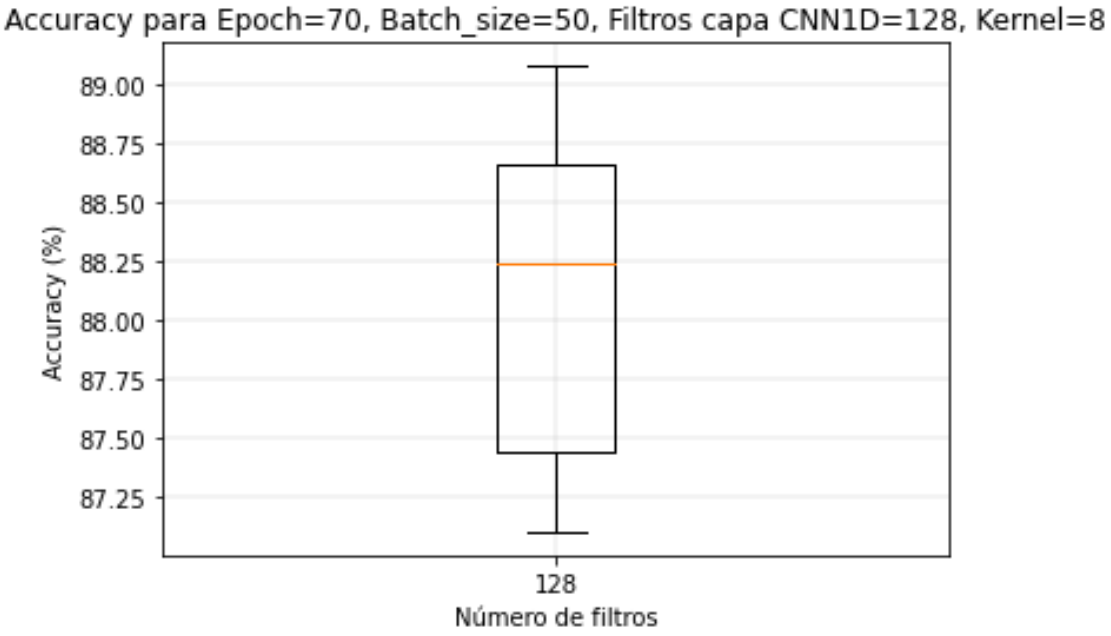


- A1.1.2 → Tres capas densamente conectadas
  - Diagrama de bigotes

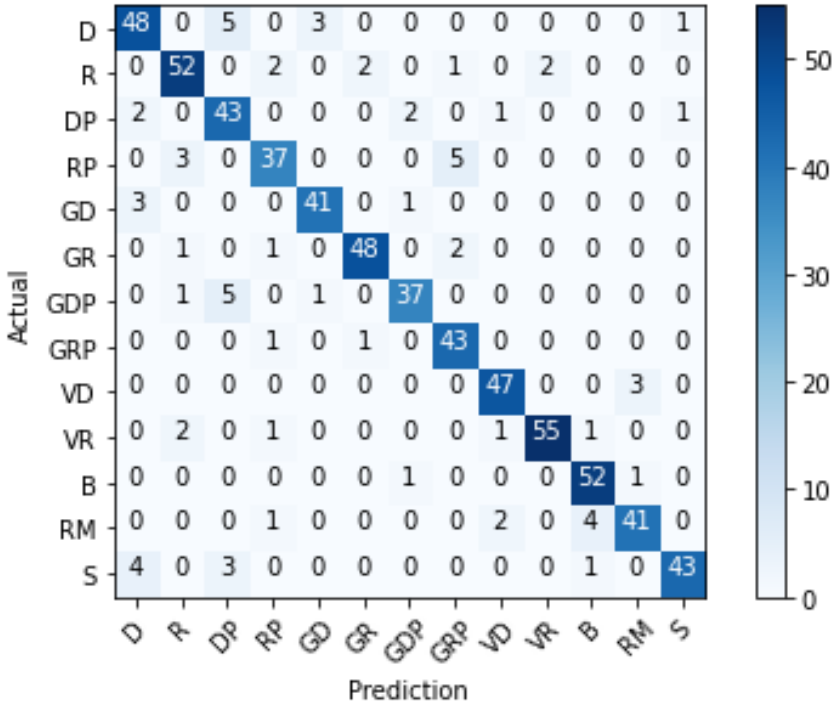


➢ A1.2 → REDES NEURONALES CONVOLUCIONALES 1D

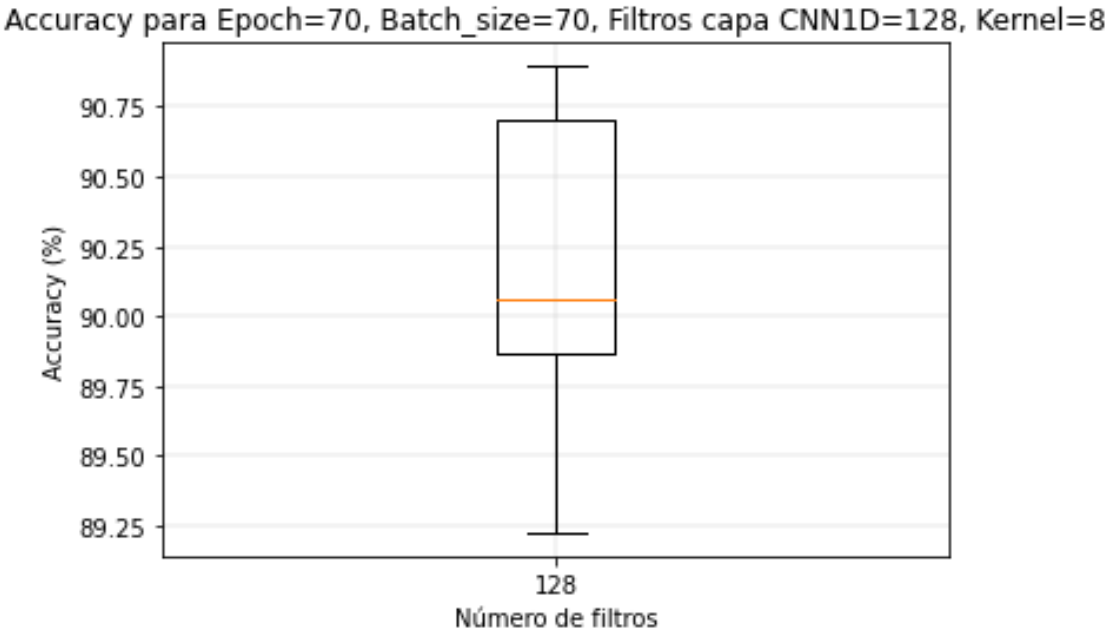
- A1.2.1 → Una capa convolucional y una capa densa
  - Diagrama de bigotes



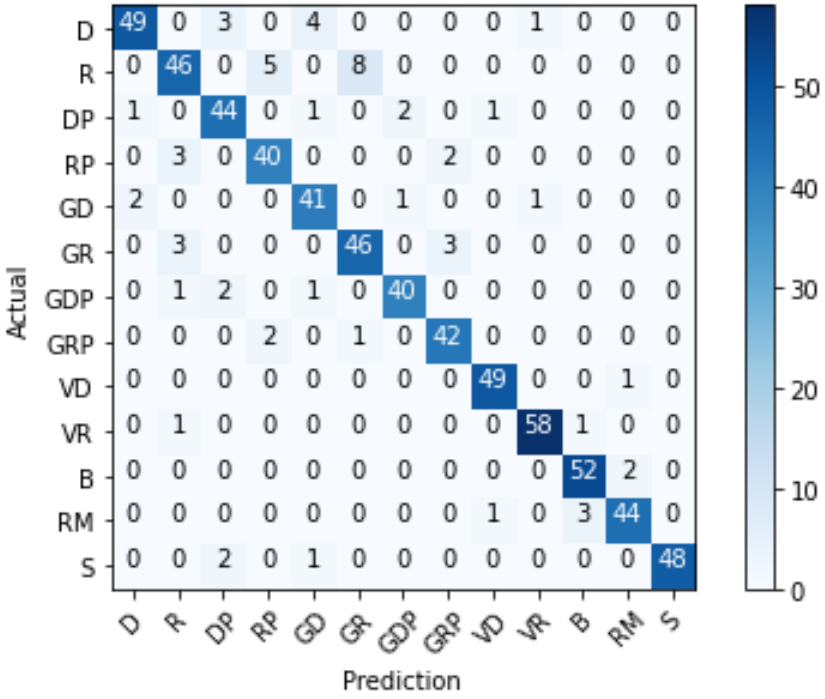
- Matriz de confusión



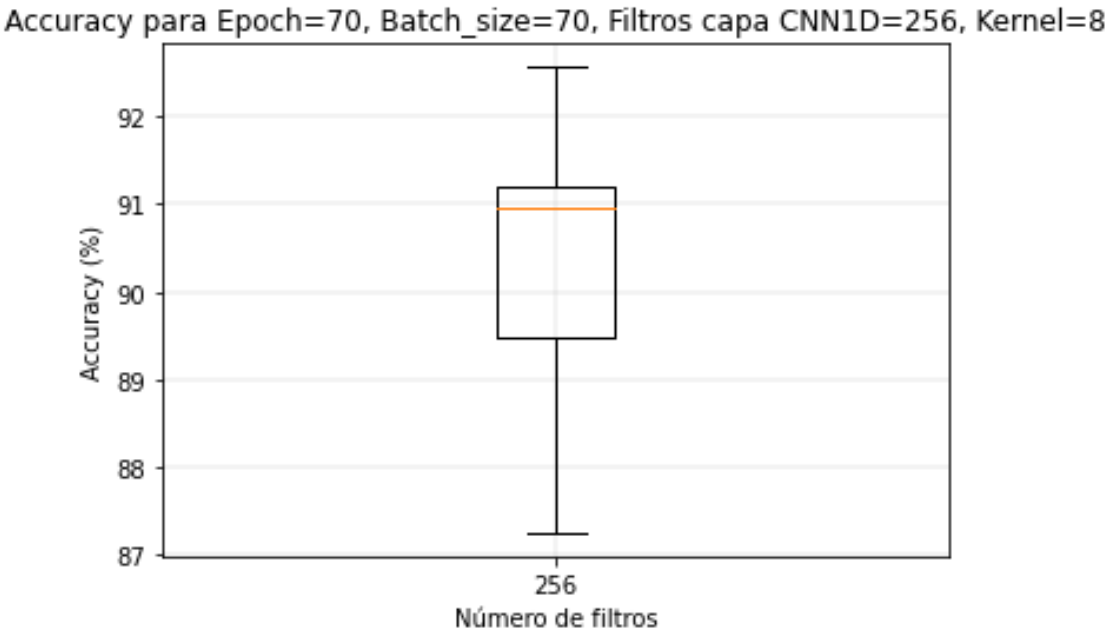
- A1.2.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



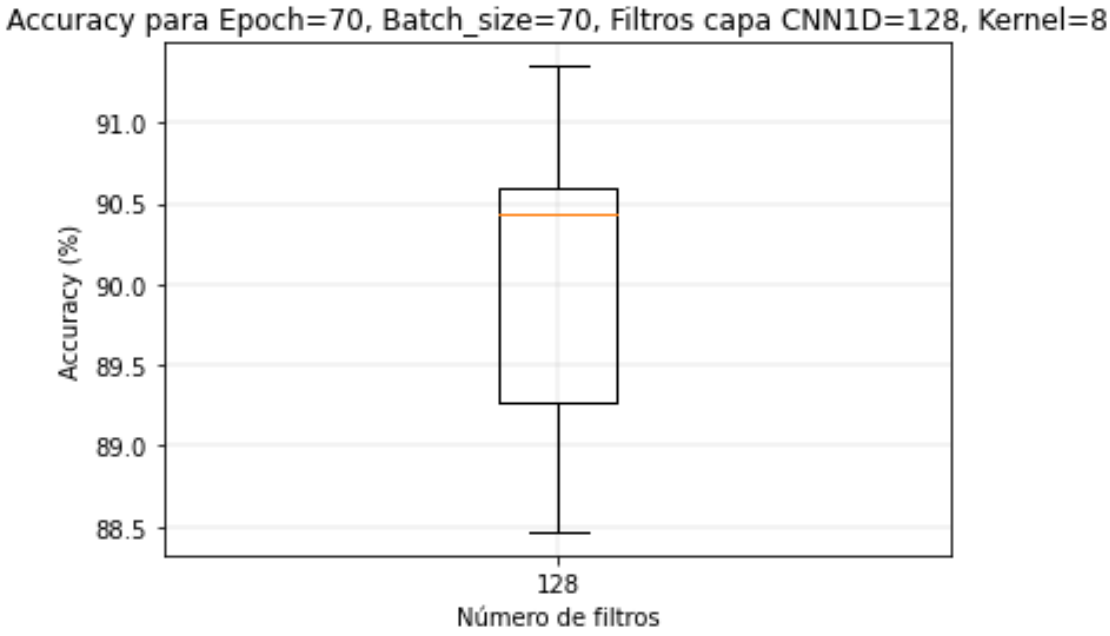
- Matriz de confusión



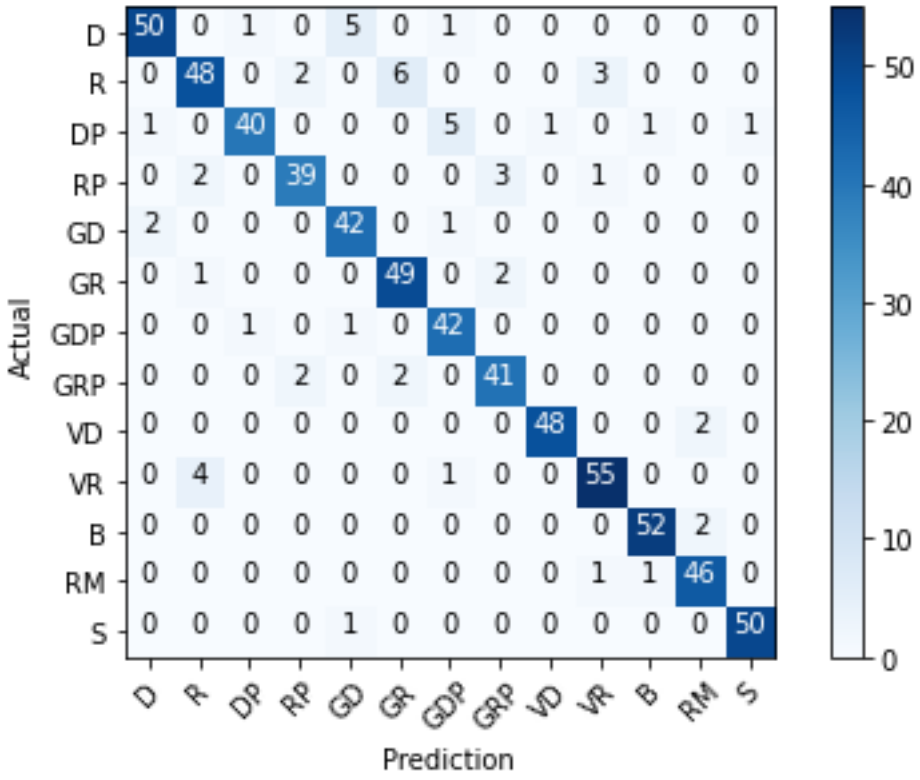
- A1.2.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



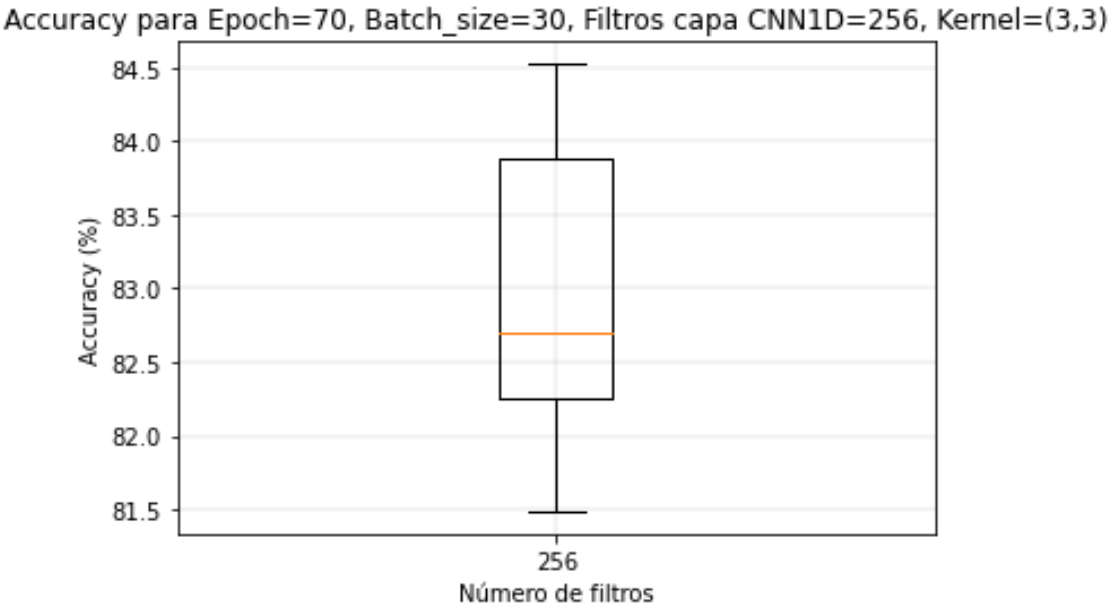
- A1.2.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



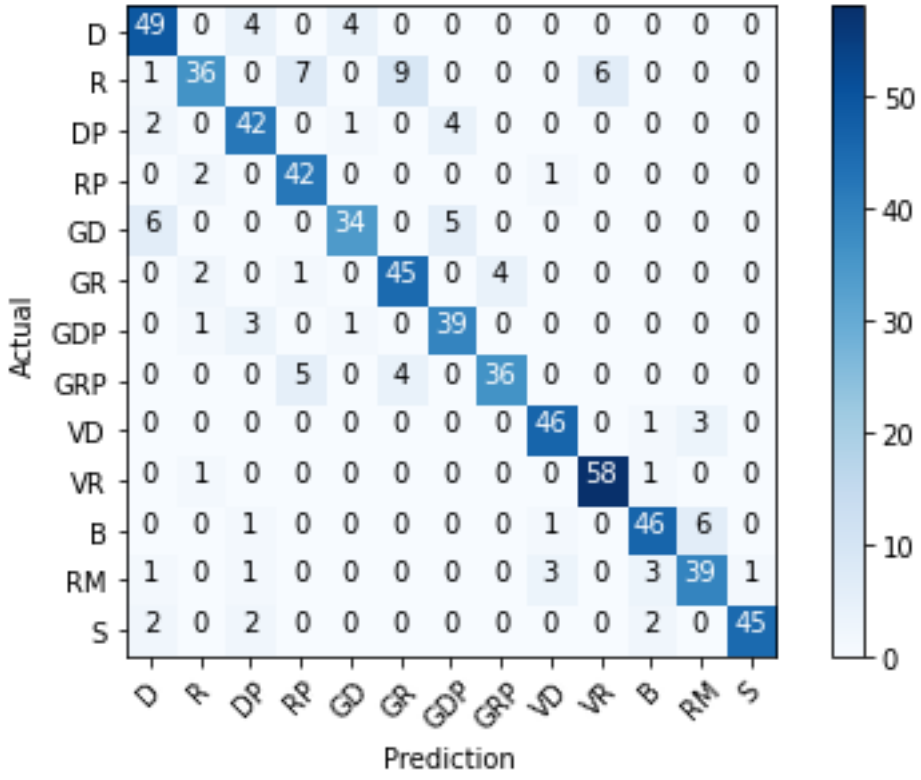
  - Matriz de confusión



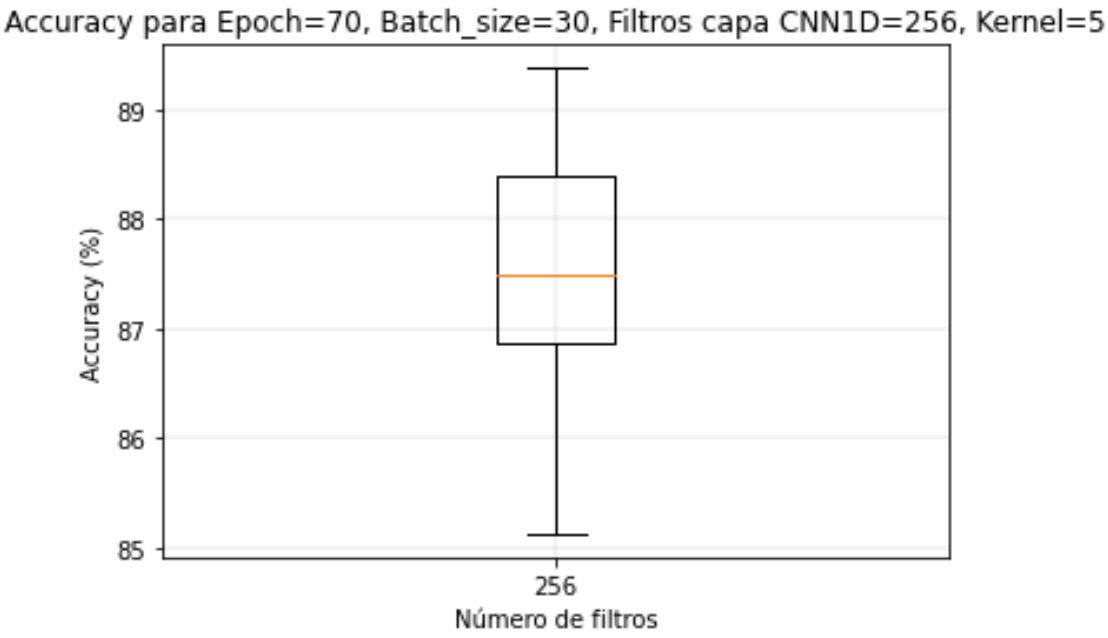
- A1.3 → REDES CONVOLUCIONALES 2D
  - A1.3.1 → Una capa convolucional y una capa densa
    - Diagrama de bigotes



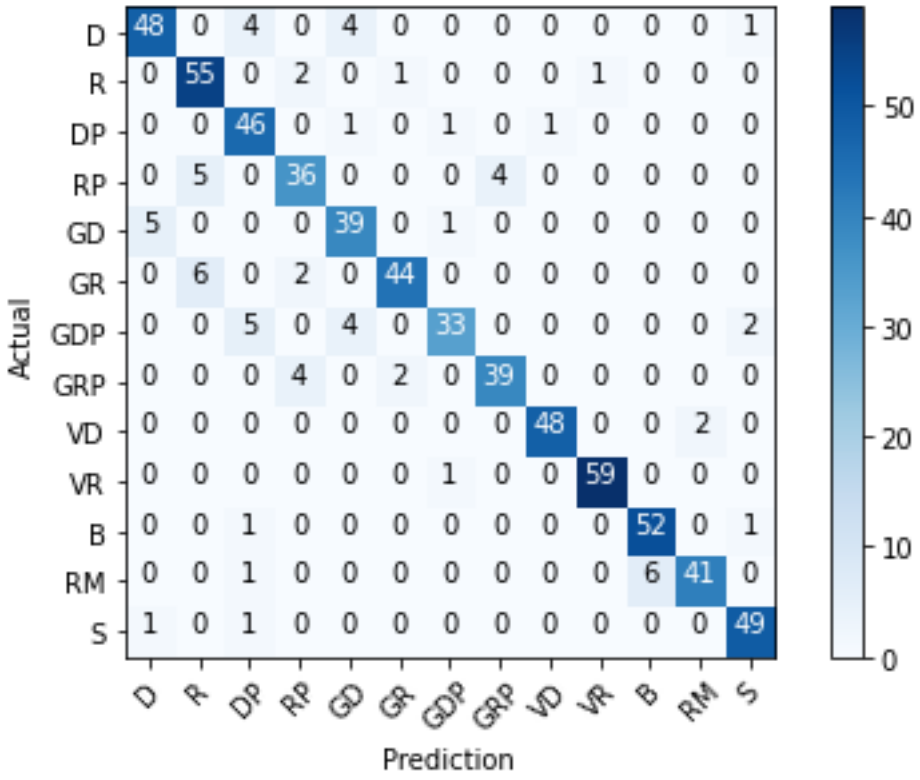
    - Matriz de confusión



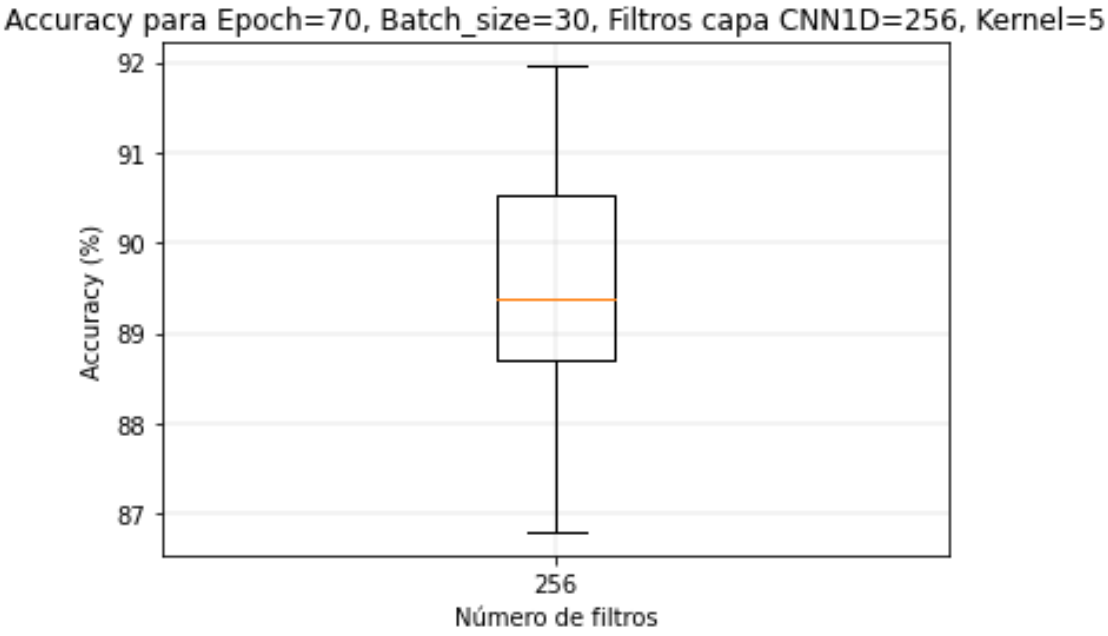
- A1.3.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



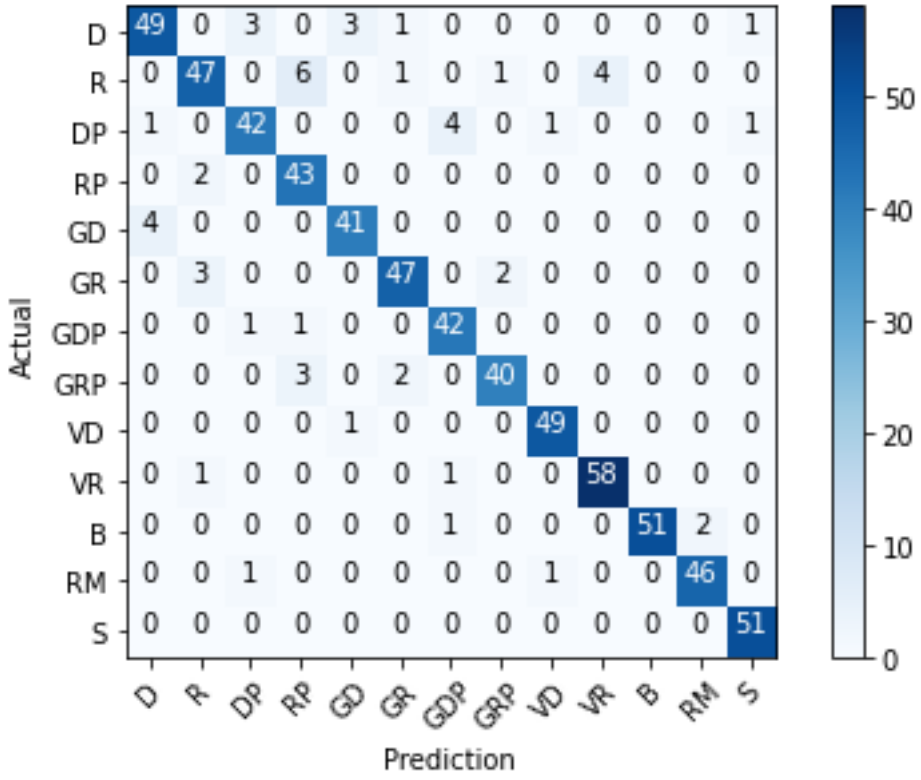
  - Matriz de confusión



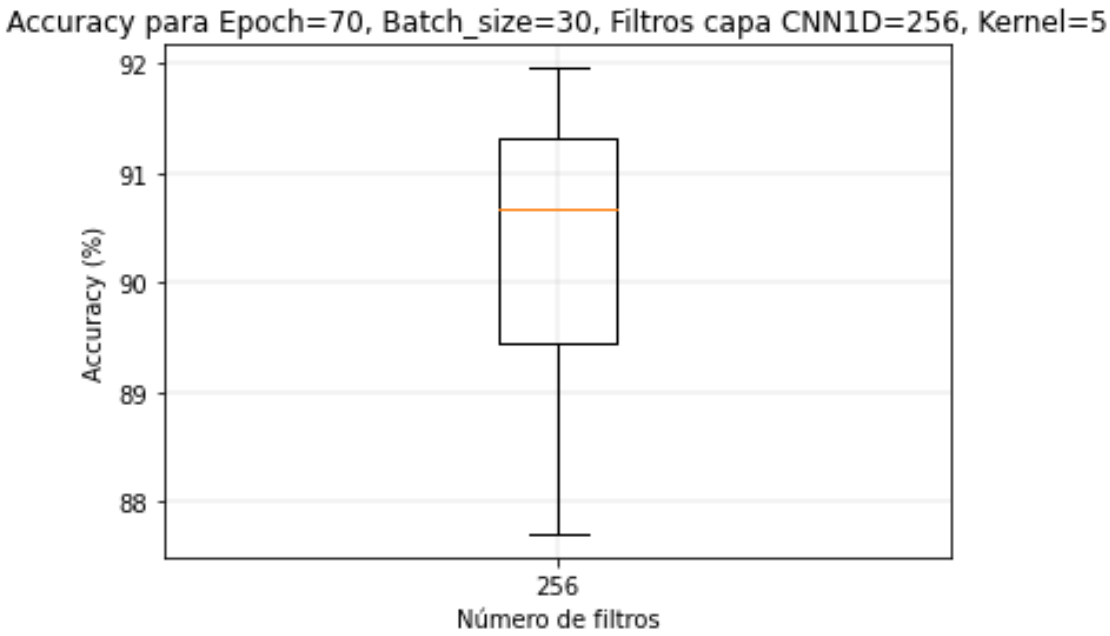
- A1.3.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



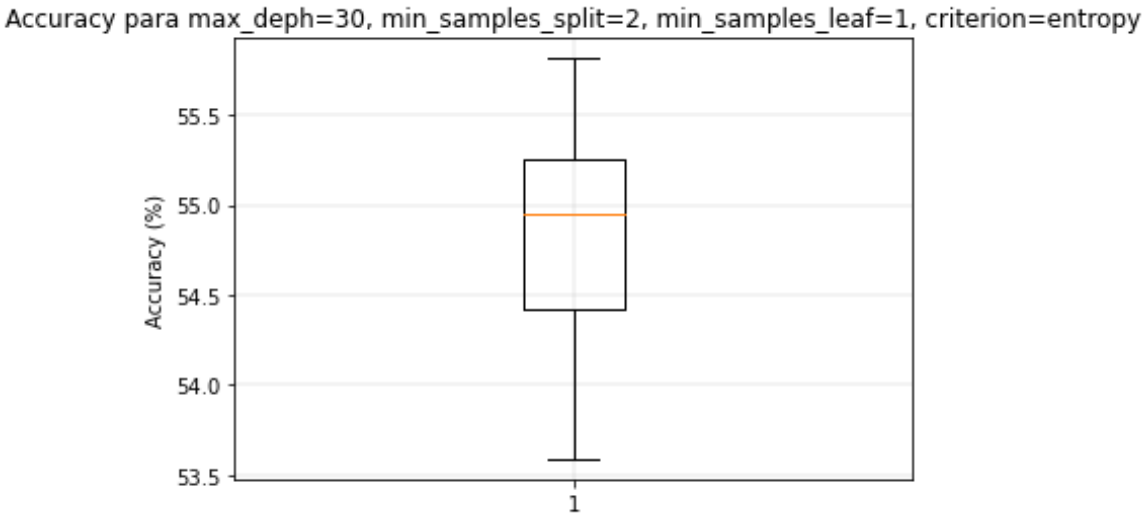
  - Matriz de confusión



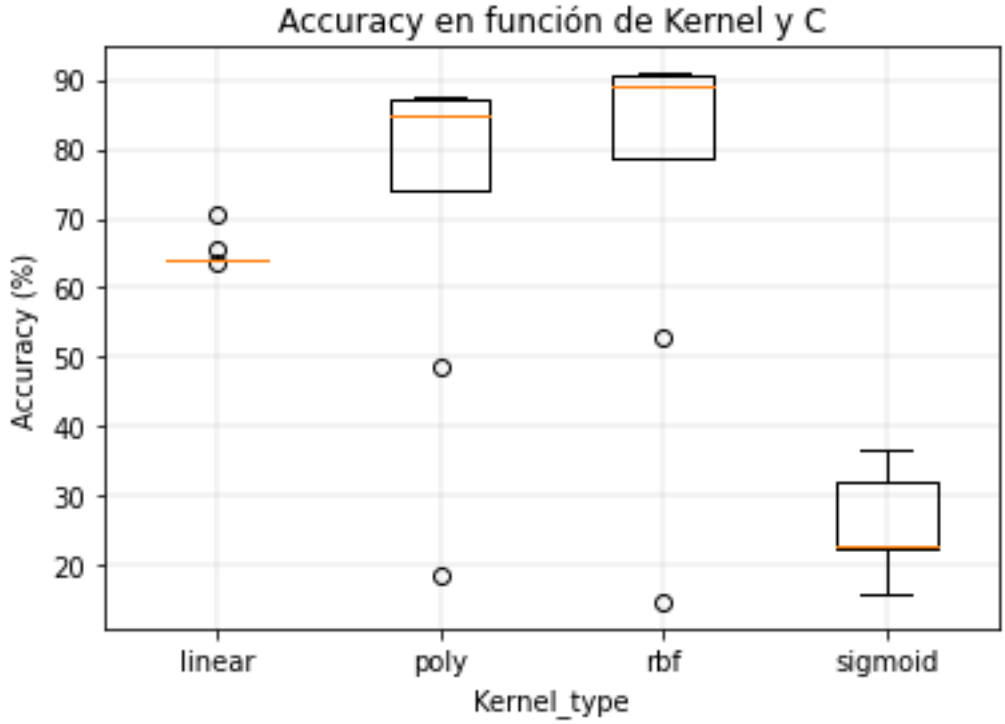
- A1.3.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



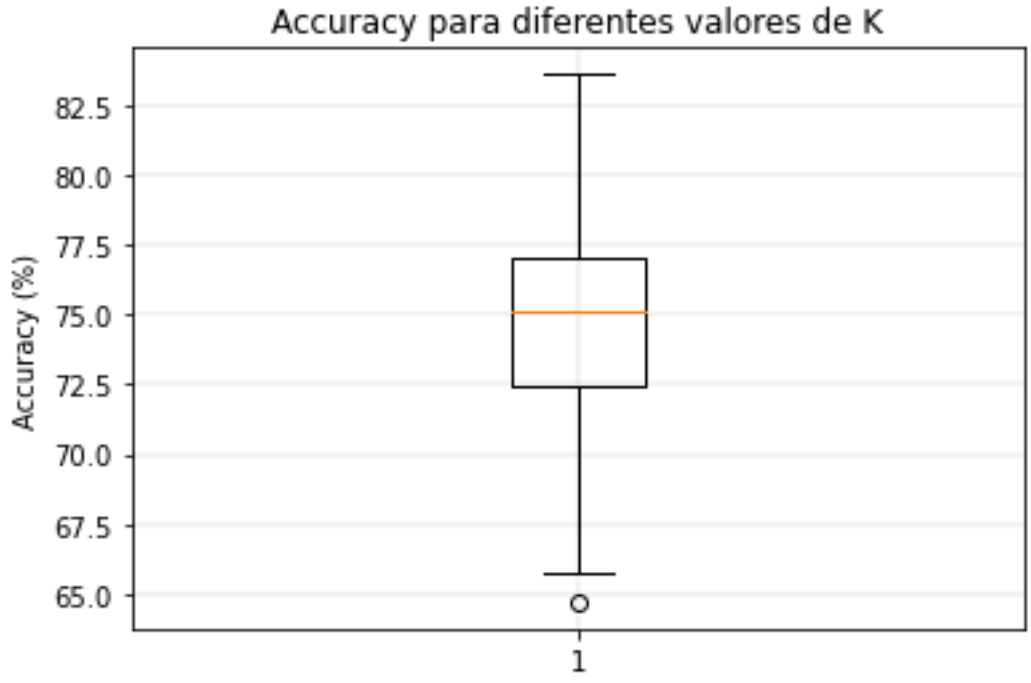
- A1.4 → ÁRBOL DE DECISIÓN
  - Diagrama de bigotes



- A1.5 → SVM
  - Diagrama de bigotes



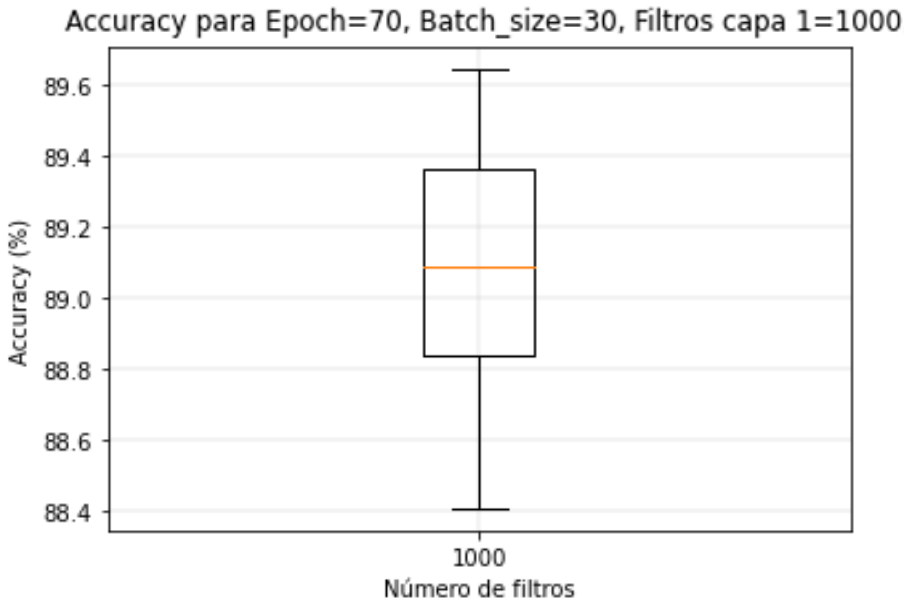
- A1.6 → KNN
  - Diagrama de bigotes



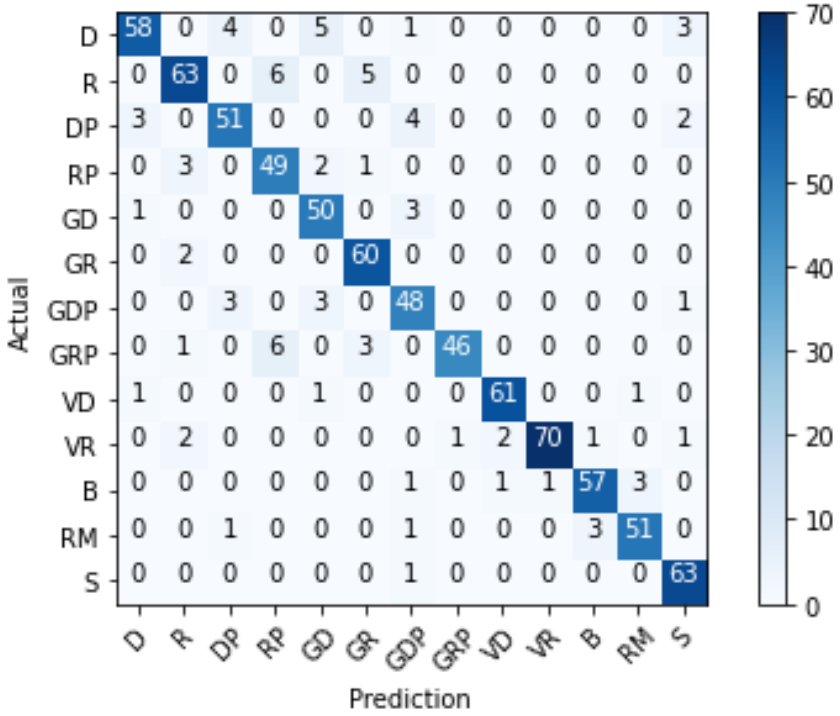
ANEXO II

A lo largo de este anexo se mostrarán los diagramas de bigotes y las matrices de confusión de las mejores configuraciones para los distintos algoritmos para el caso de estudio en el que se entrena la red neuronal con la base de datos en el dominio del tiempo y con jugadores zurdos y diestros.

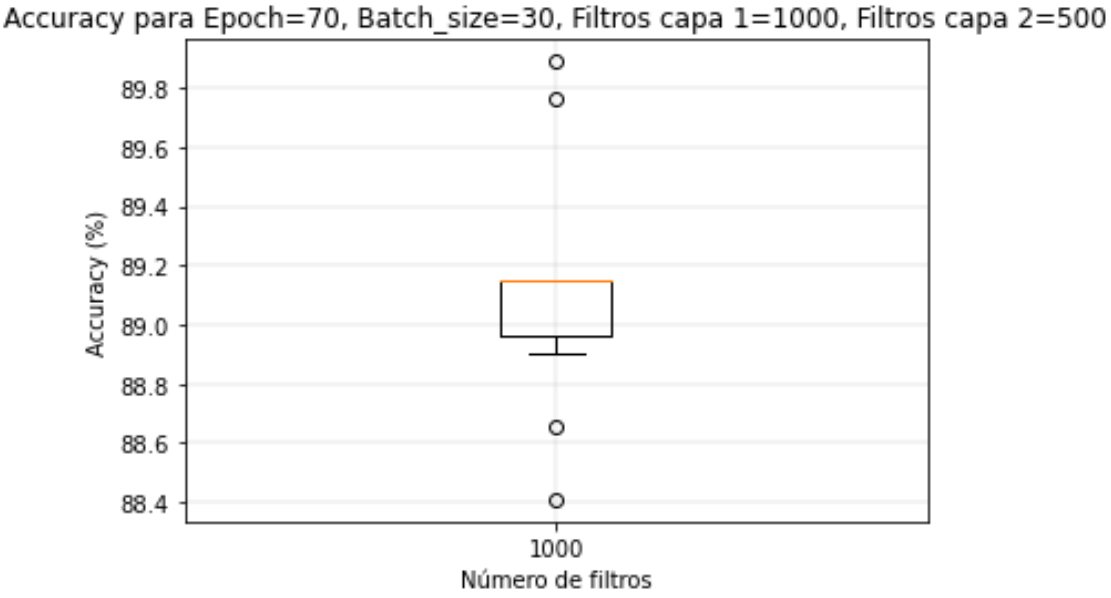
- A2.1 → REDES NEURONALES DENSAMENTE CONECTADAS
  - A2.1.1 → Dos capas densamente conectadas
    - Diagrama de bigotes



    - Matriz de confusión

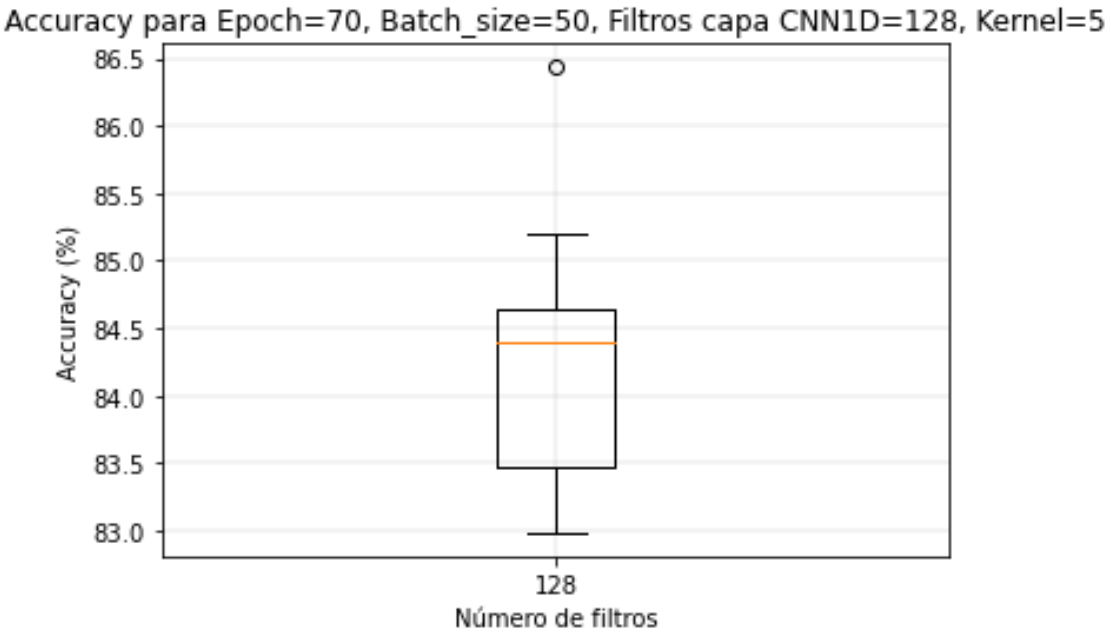


- A2.1.2 → Tres capas densamente conectadas
  - Diagrama de bigotes

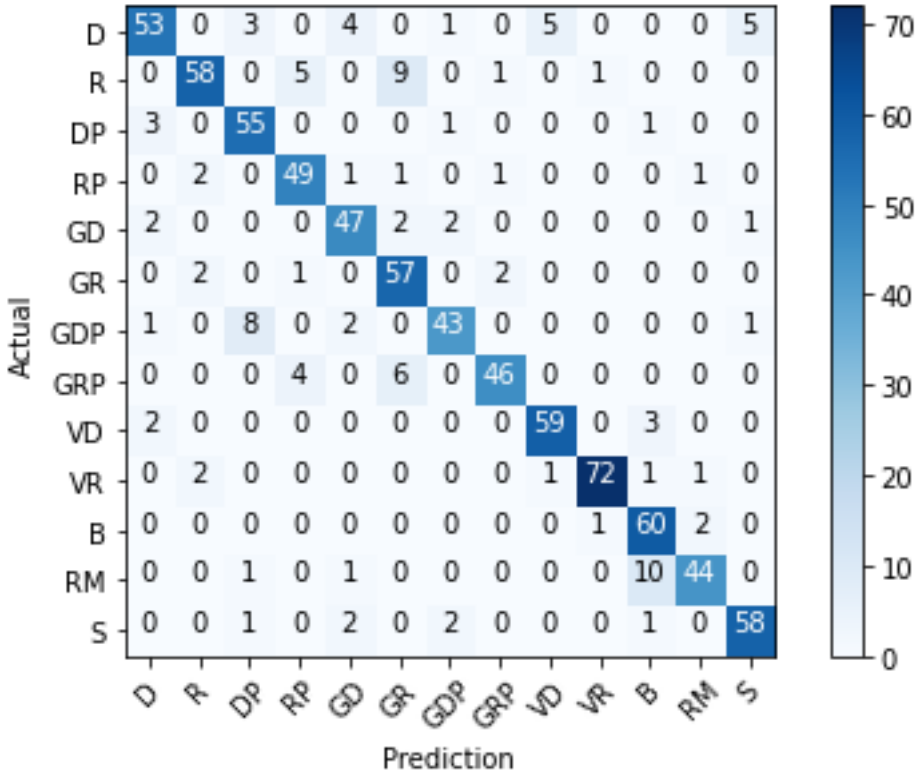


➢ A2.2 → REDES NEURONALES CONVOLUCIONALES

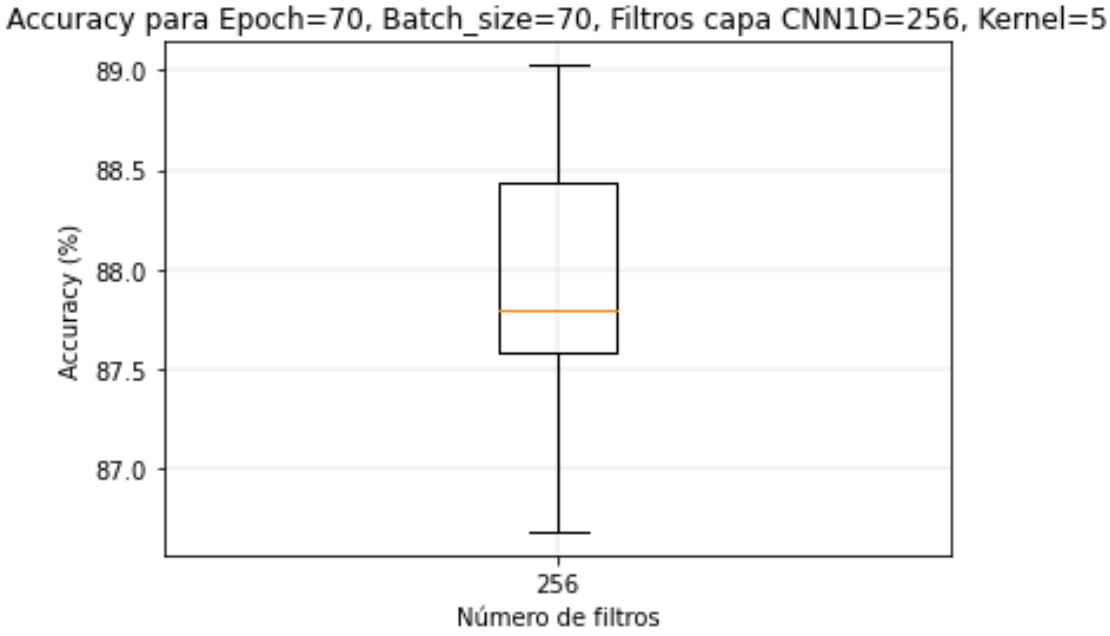
- A2.2.1 → Una capa convolucional y una capa densa
  - Diagrama de bigotes



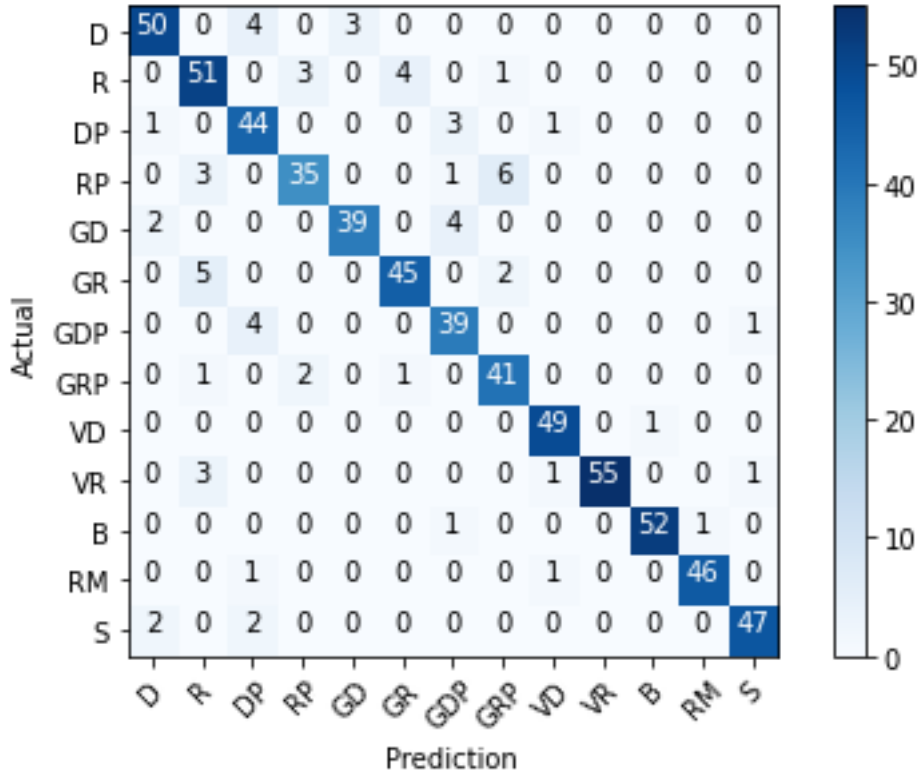
- Matriz de confusión



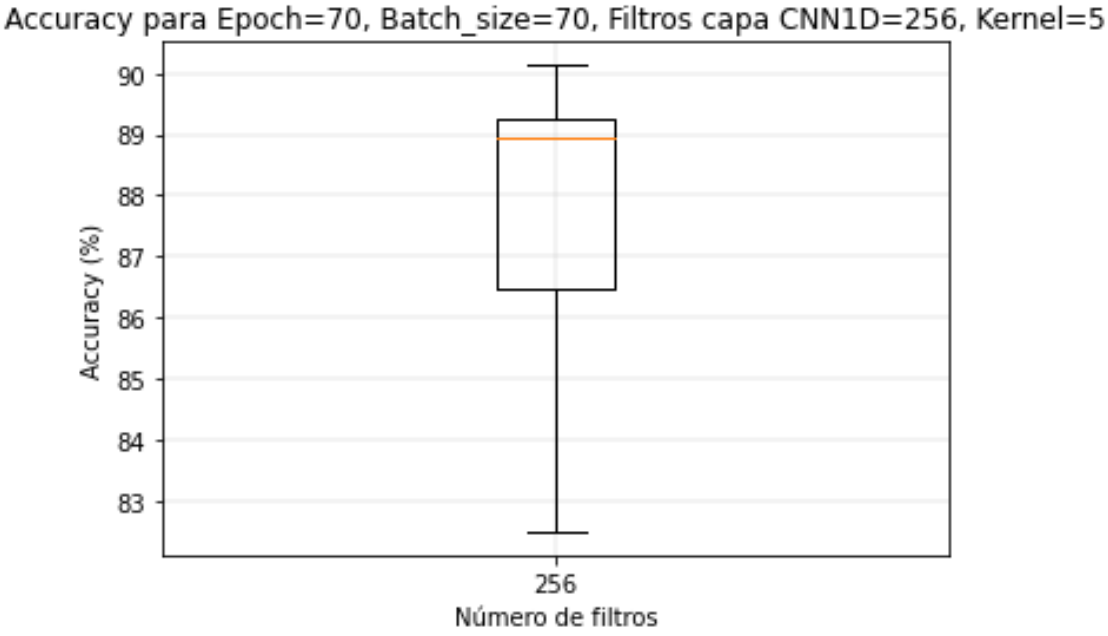
- A2.2.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



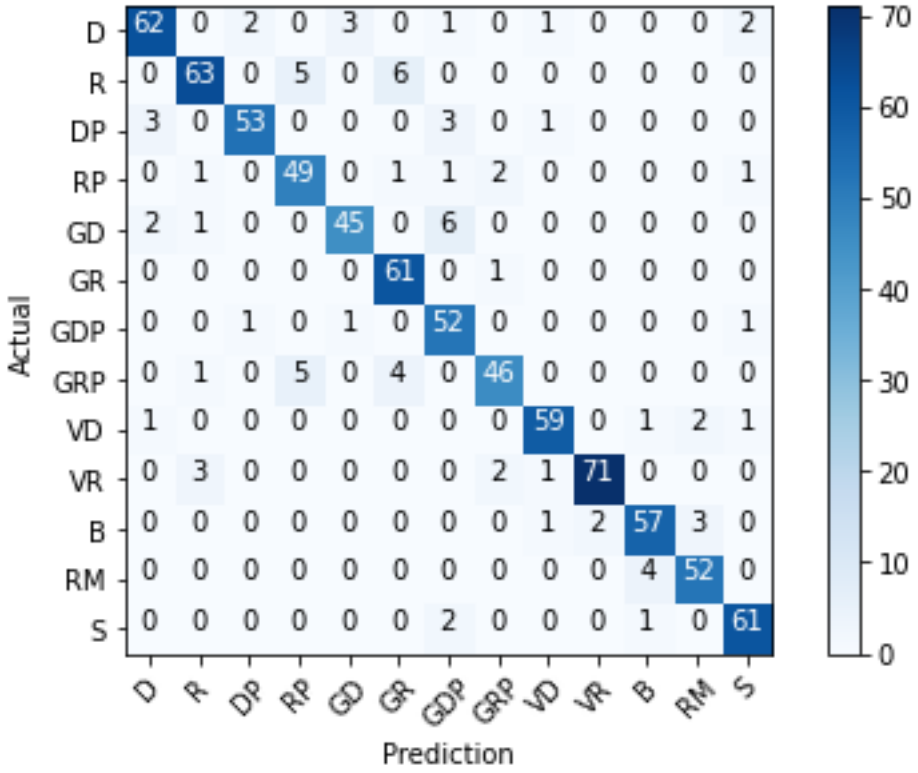
- Matriz de confusión



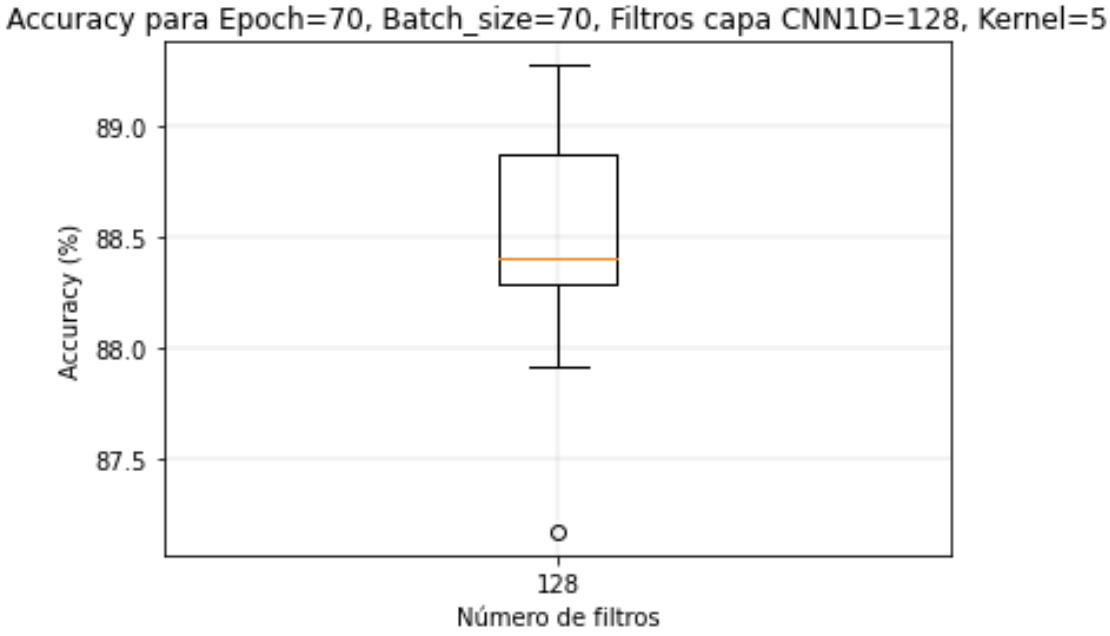
- A2.2.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



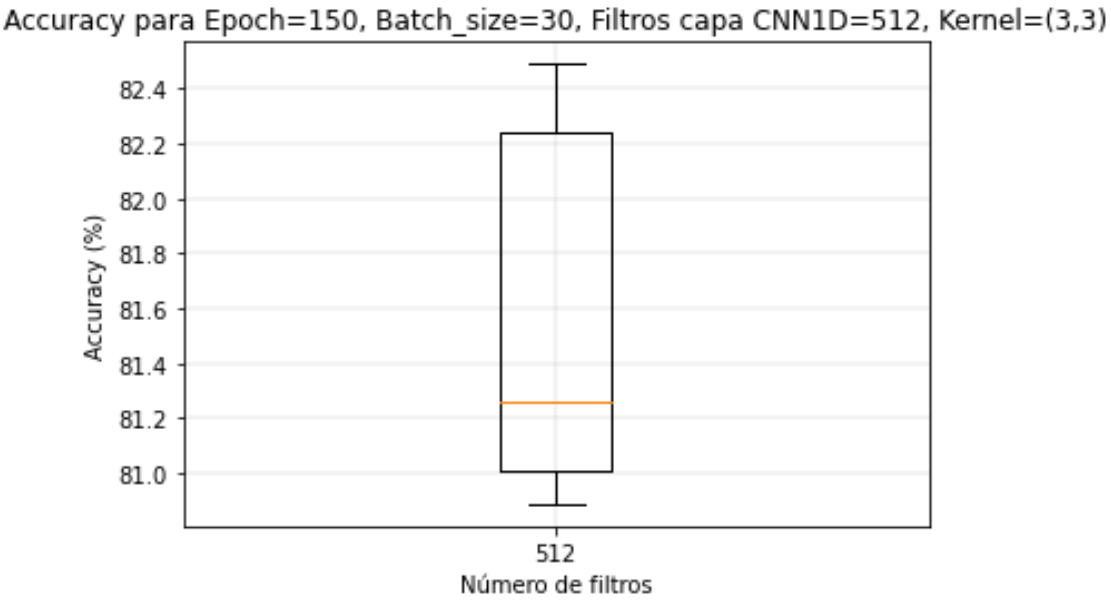
- Matriz de confusión



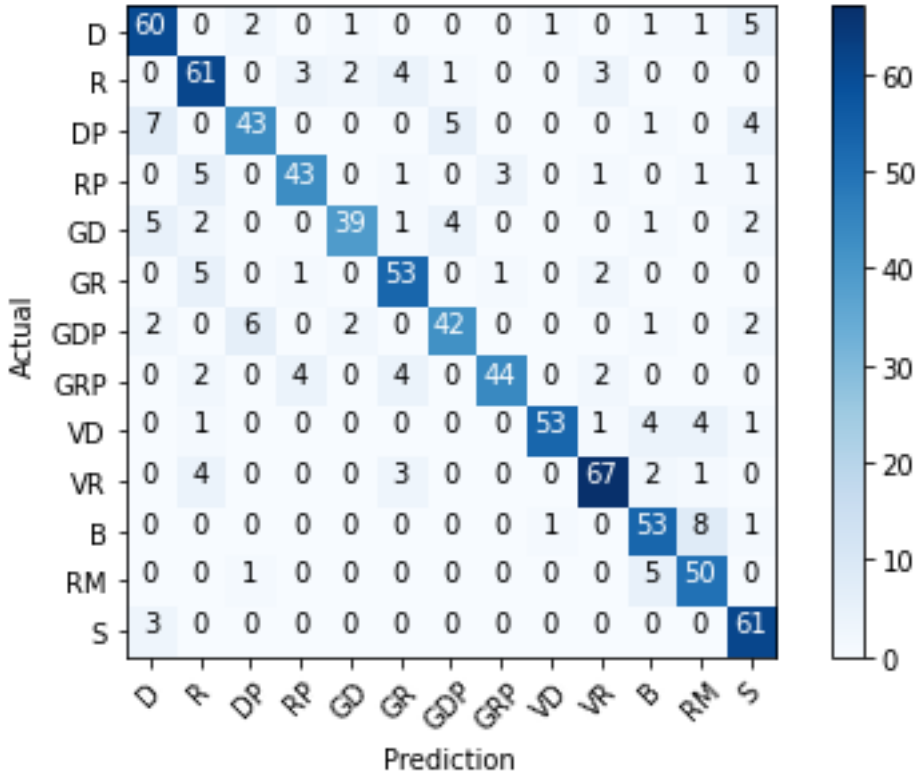
- A2.2.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



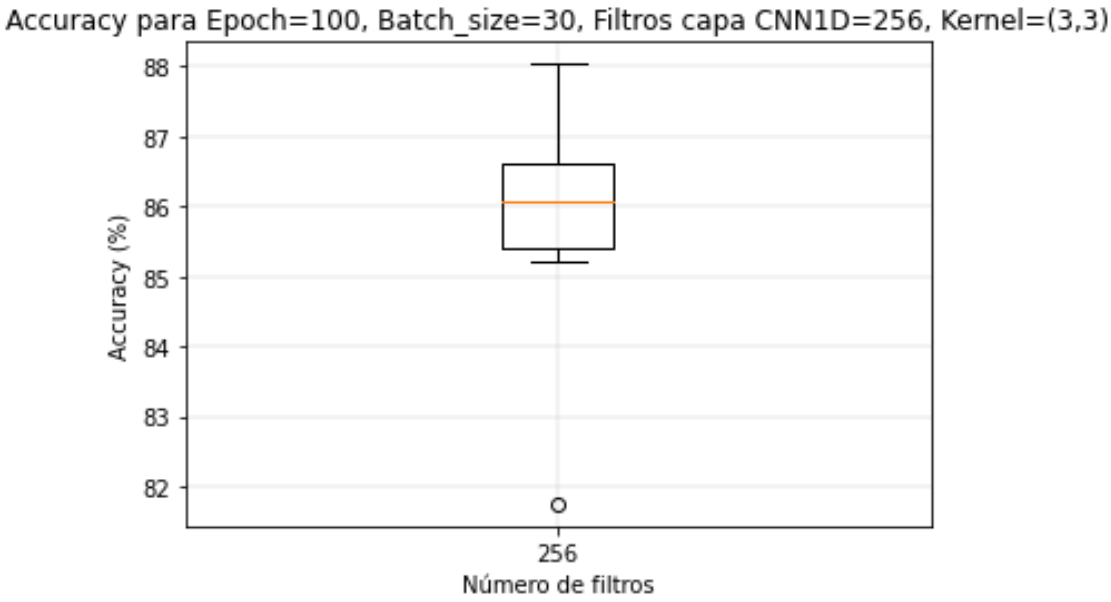
- A2.3 → REDES CONVOLUCIONALES 2D
  - A2.3.1 → Una capa convolucional y una capa densa
    - Diagrama de bigotes



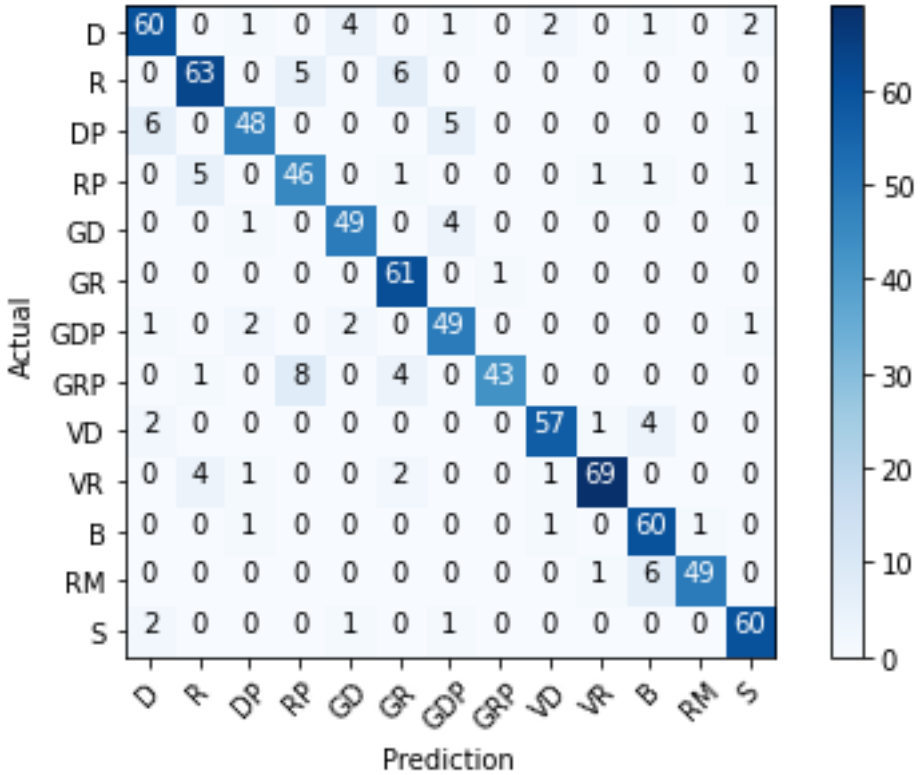
    - Matriz de confusión



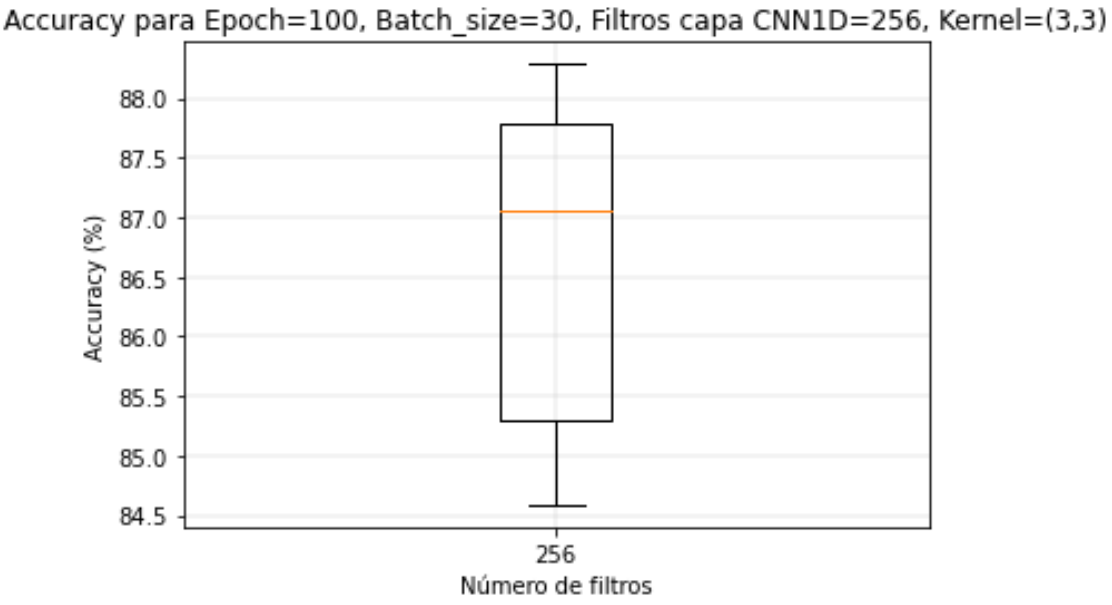
- A2.3.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



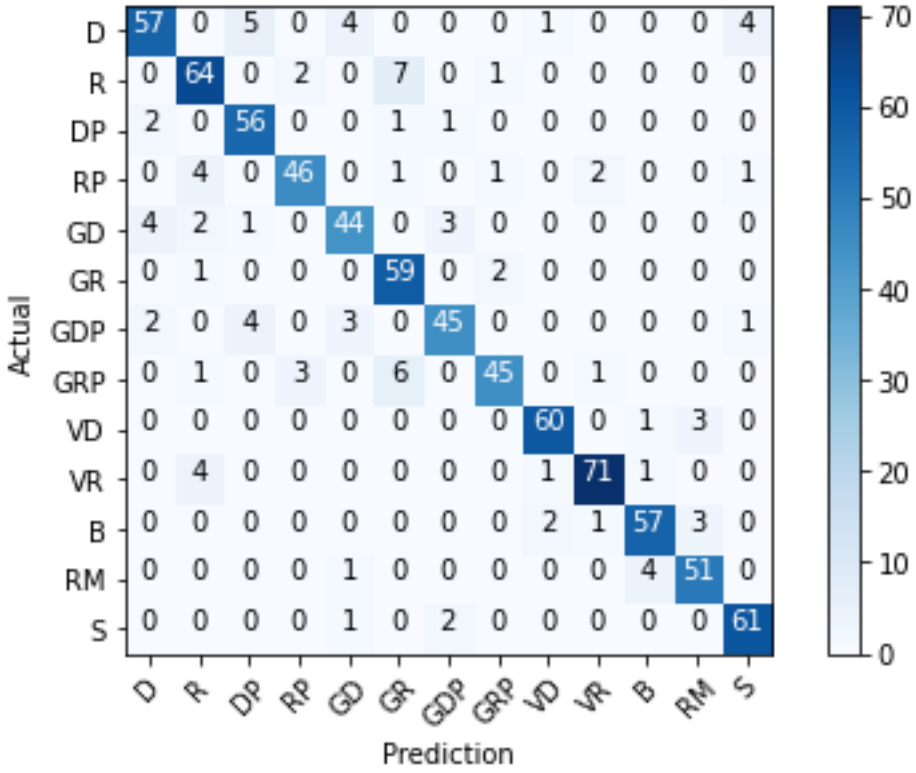
  - Matriz de confusión



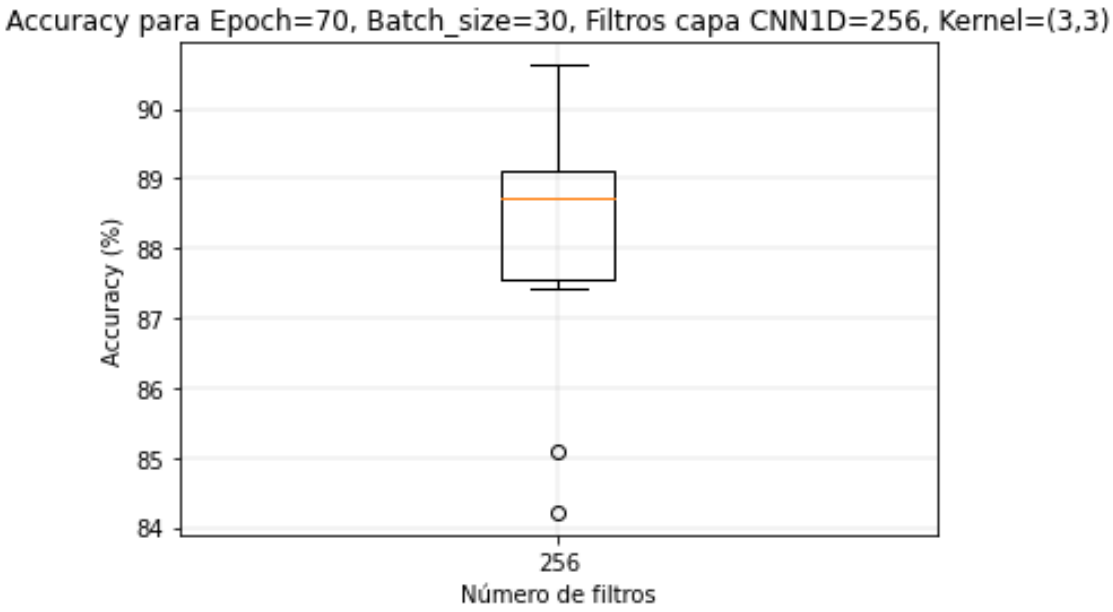
- A2.3.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



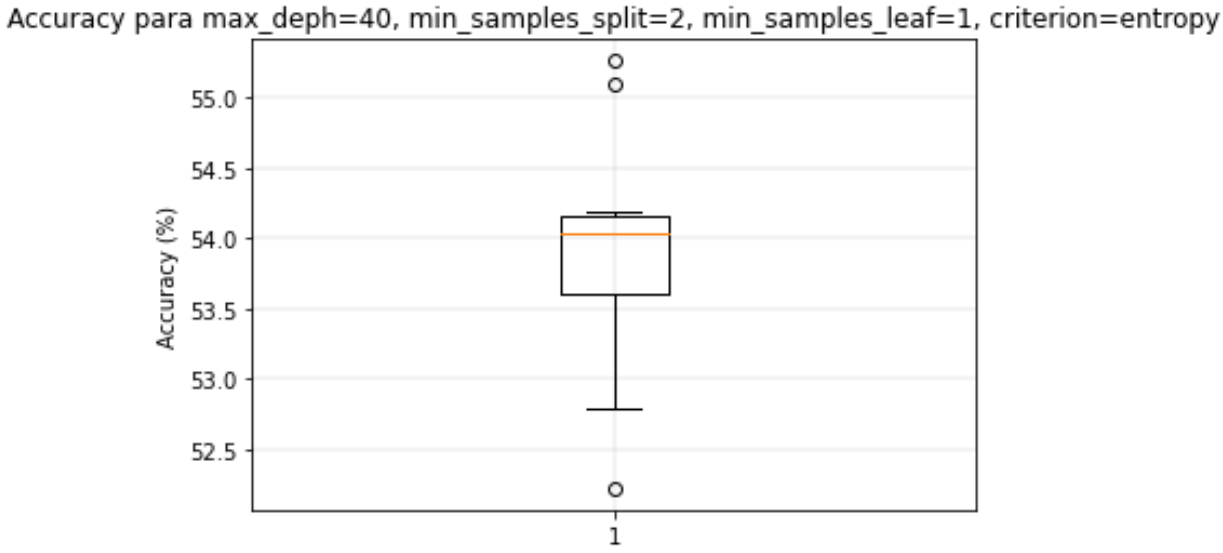
  - Matriz de confusión



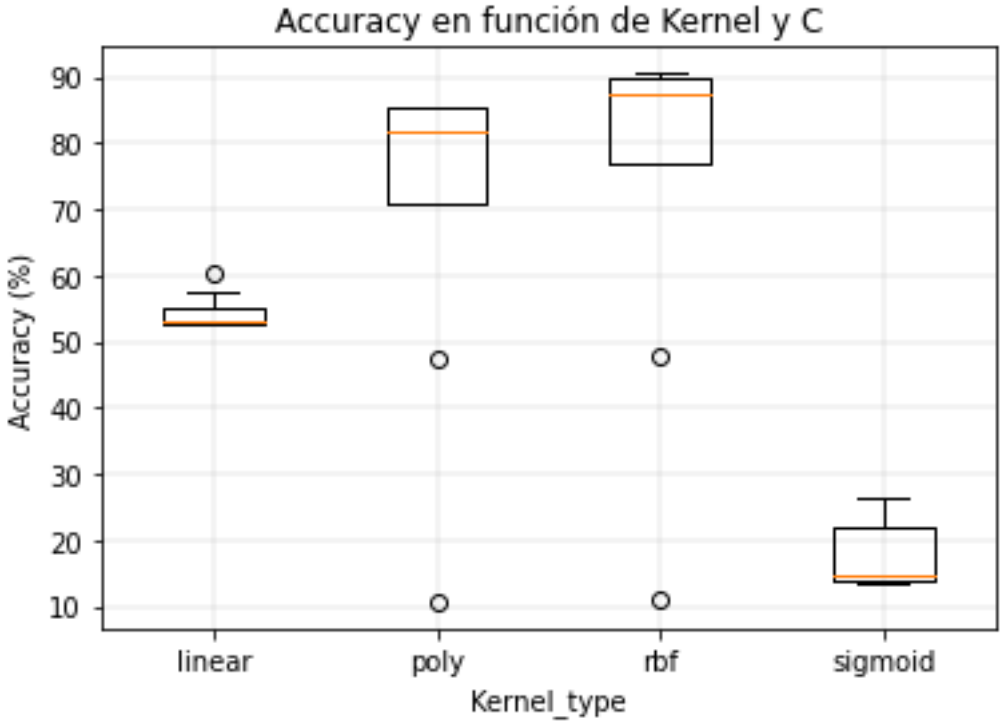
- A2.3.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



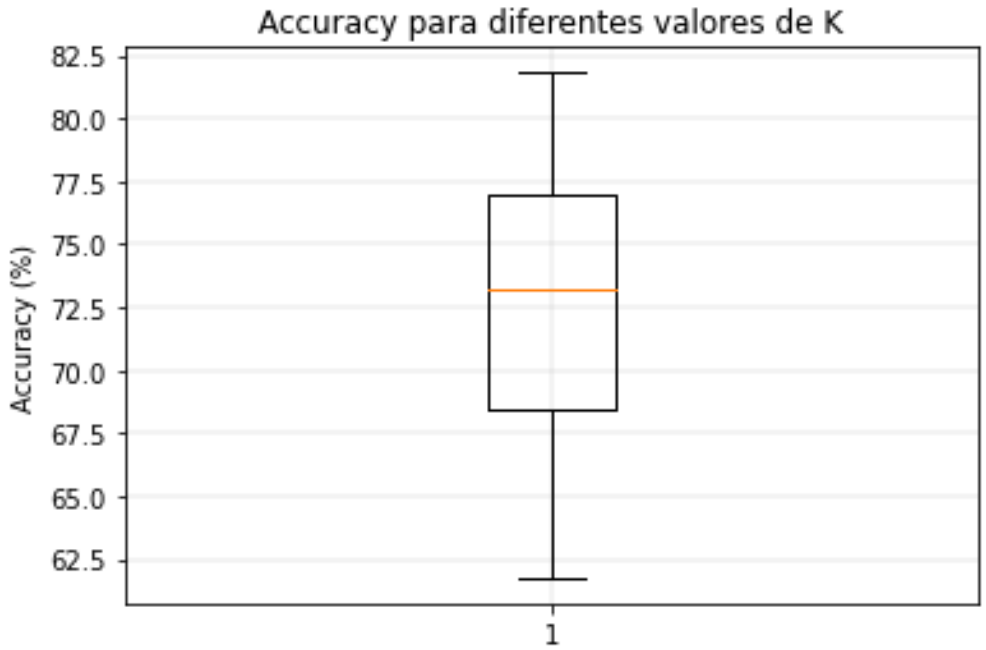
- A2.4→ ÁRBOL DE DECISIÓN
  - Diagrama de bigotes



- A2.5 → SVM
  - Diagrama de bigotes



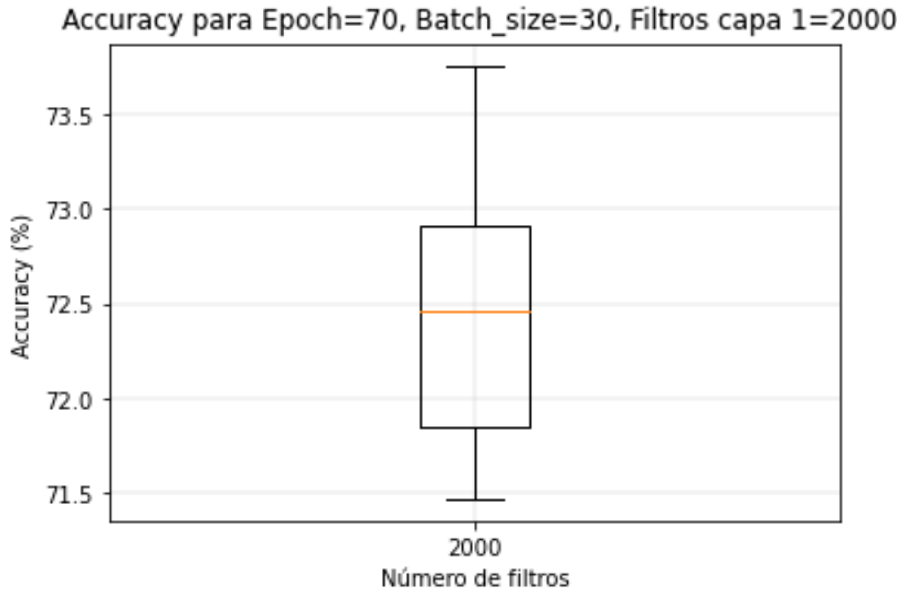
- A2.6 → KNN
  - Diagrama de bigotes



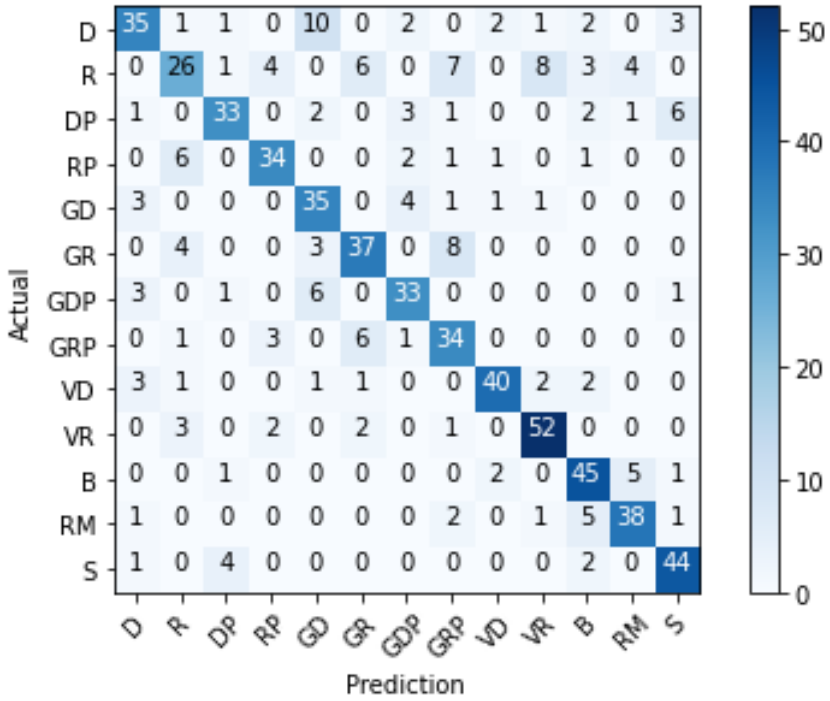
ANEXO III

A lo largo de este anexo se mostrarán los diagramas de bigotes y las matrices de confusión de las mejores configuraciones para los distintos algoritmos para el caso de estudio en el que se entrena la red neuronal con la base de datos en el dominio de la frecuencia y sin jugadores zurdos.

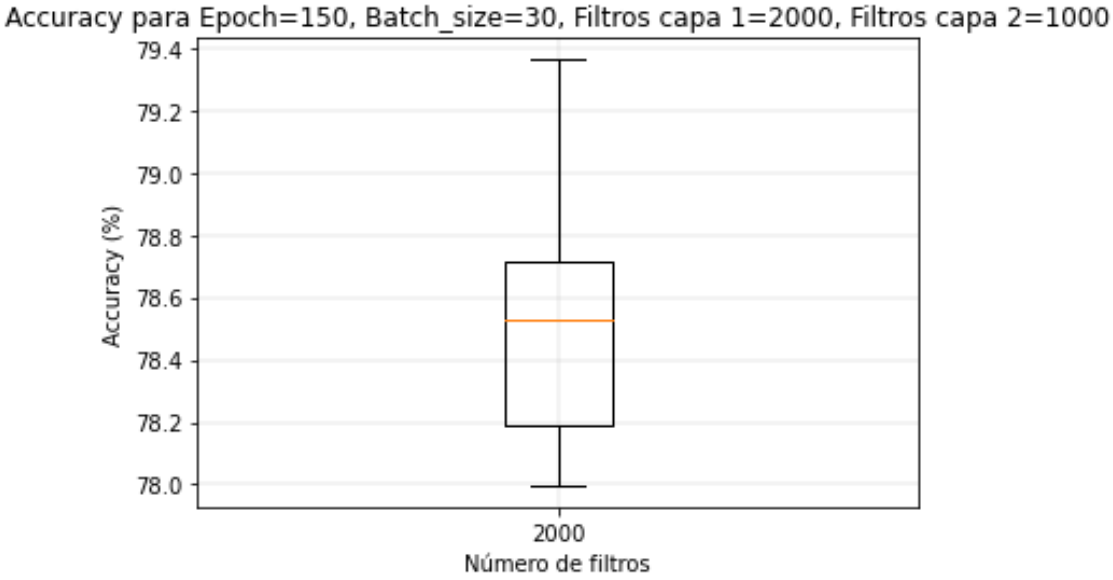
- A3.1 → REDES NEURONALES DENSAMENTE CONECTADAS
  - A3.1.1 → Dos capas densamente conectadas
    - Diagrama de bigotes



    - Matriz de confusión

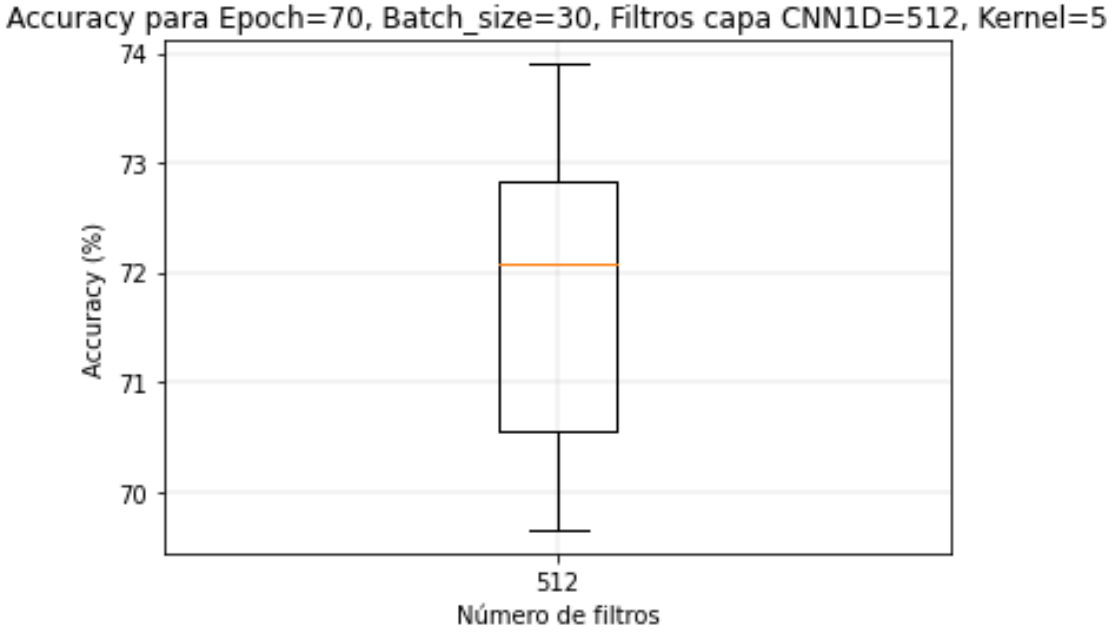


- A3.1.2 → Tres capas densamente conectadas
  - Diagrama de bigotes

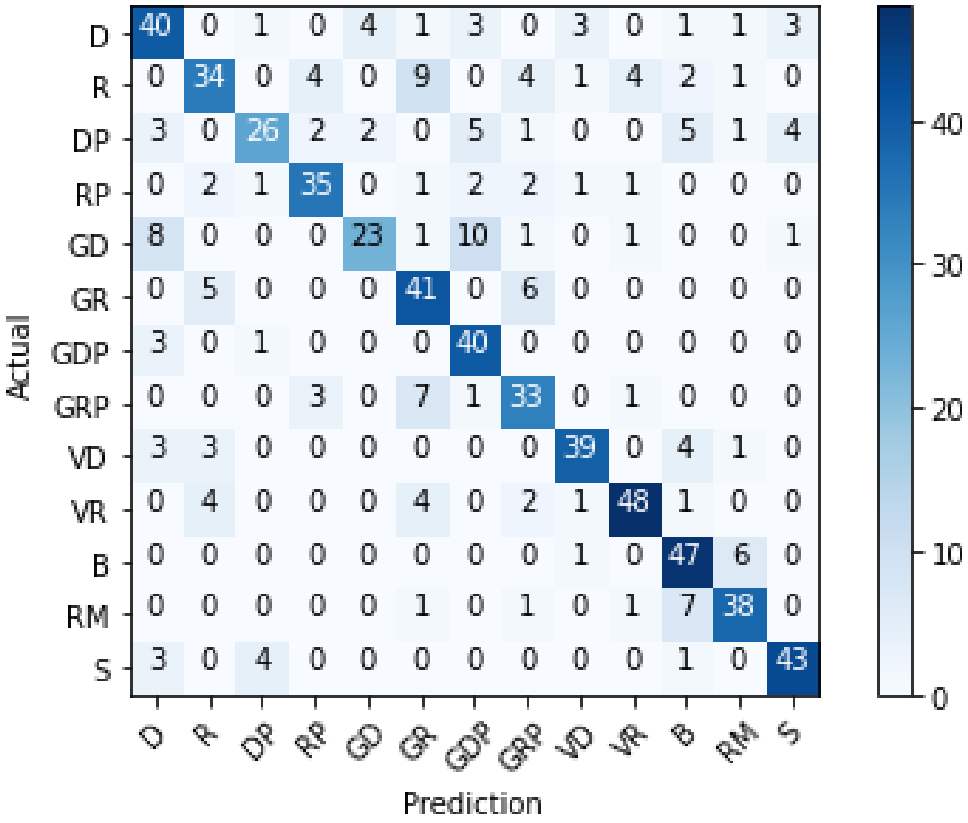


➢ A3.2 → REDES NEURONALES CONVOLUCIONALES

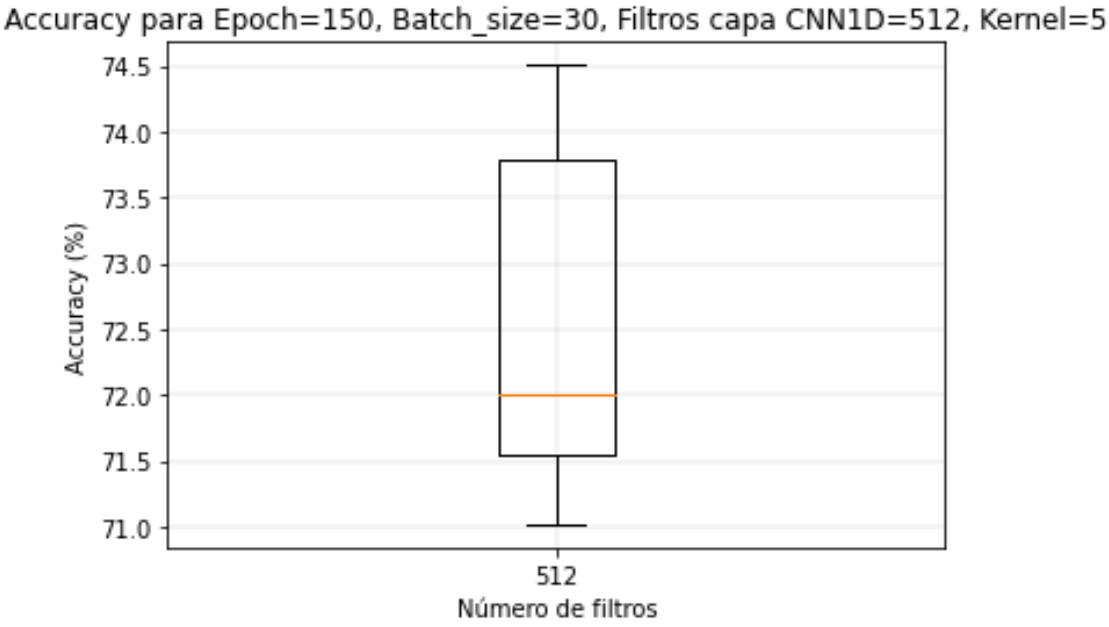
- A3.2.1 → Una capa convolucional y una capa densa
  - Diagrama de bigotes



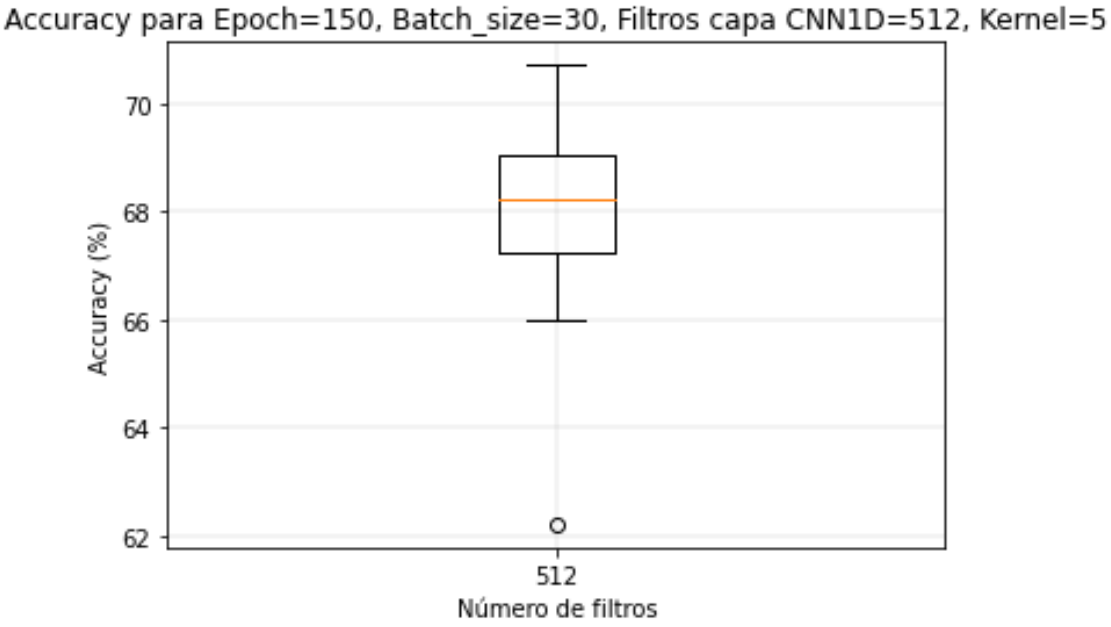
- Matriz de confusión



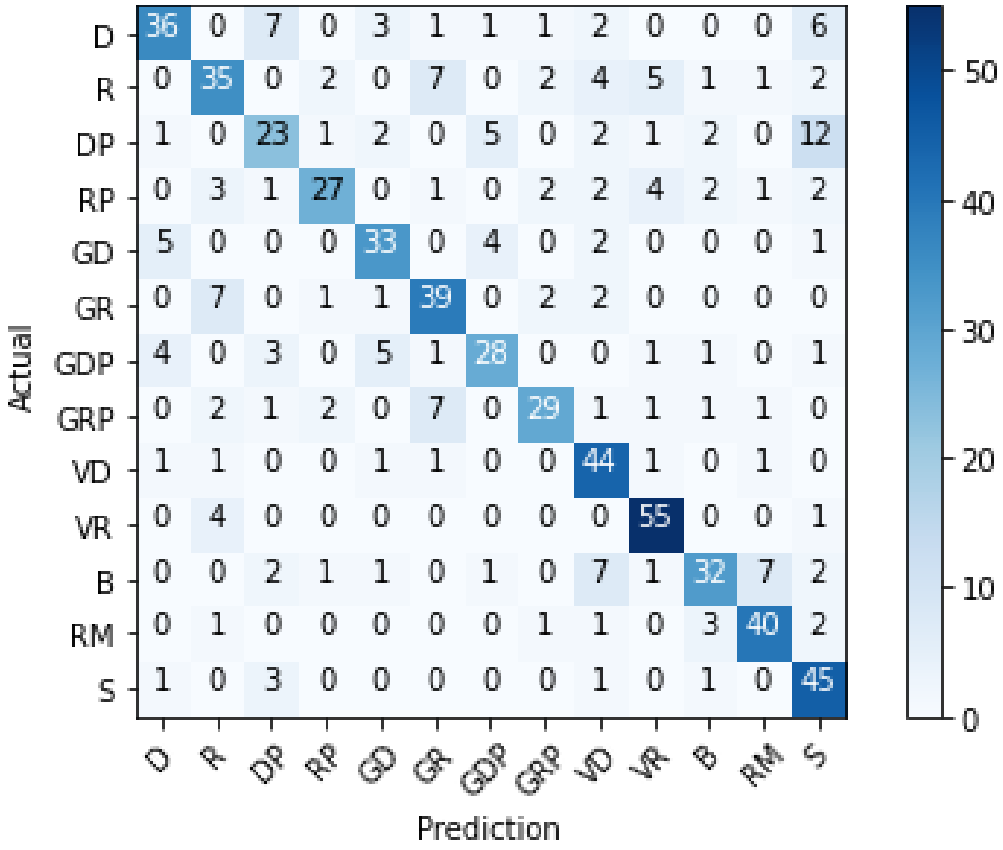
- A3.2.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



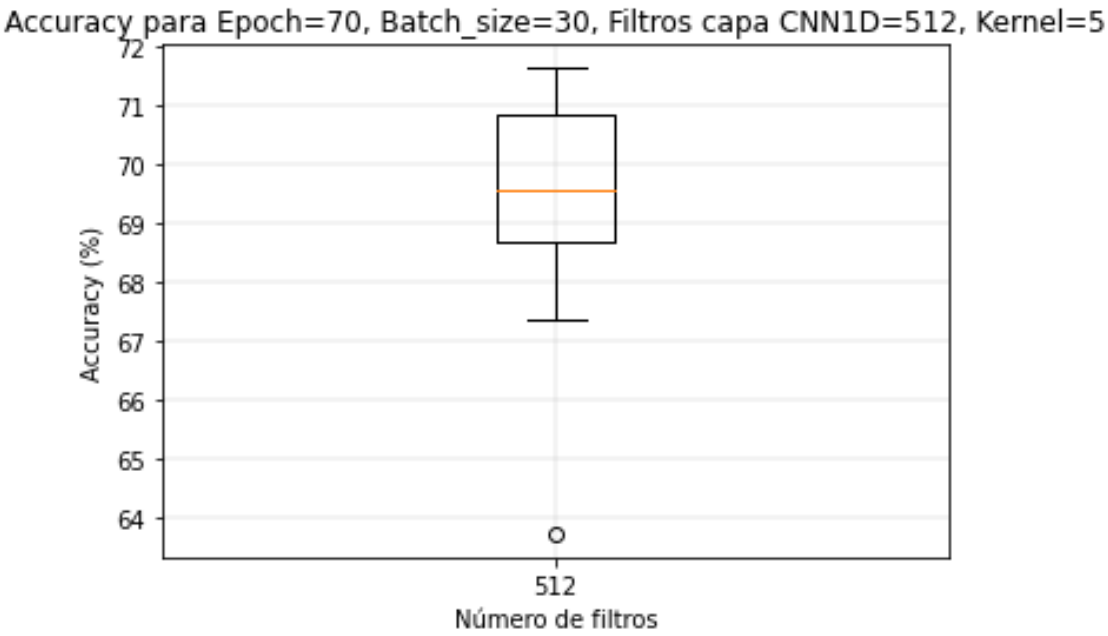
- A3.2.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



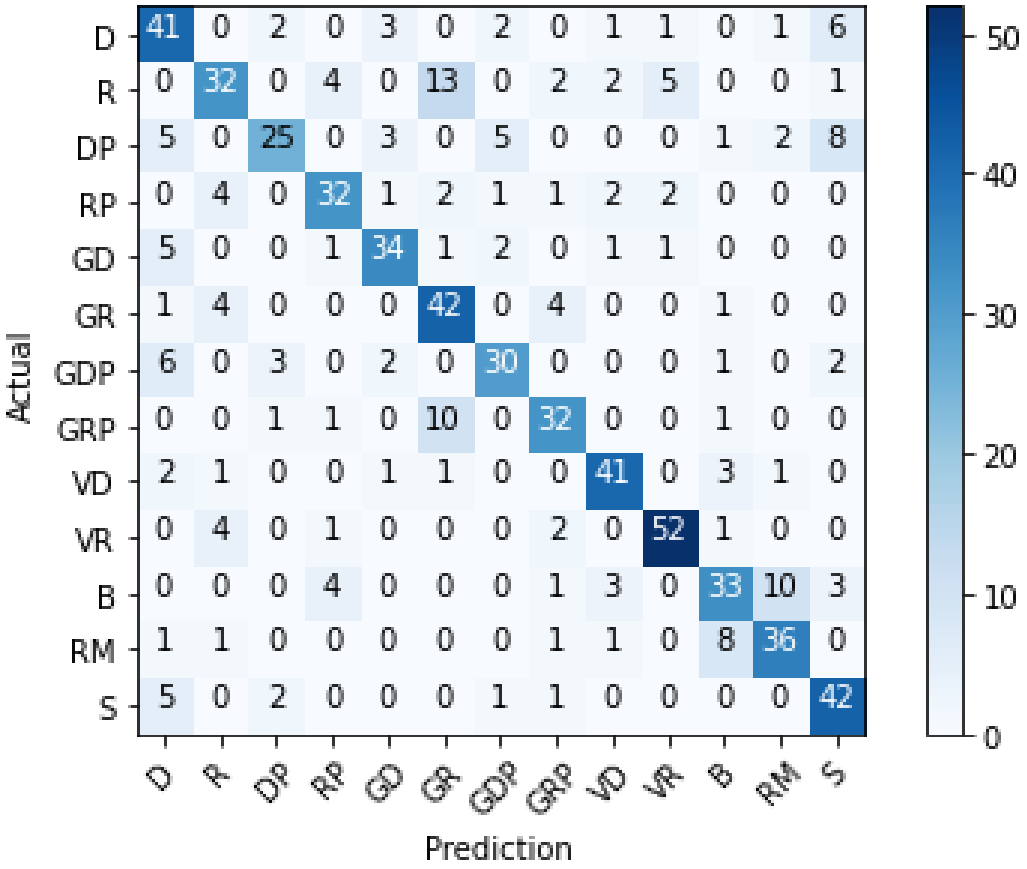
  - Matriz de confusión



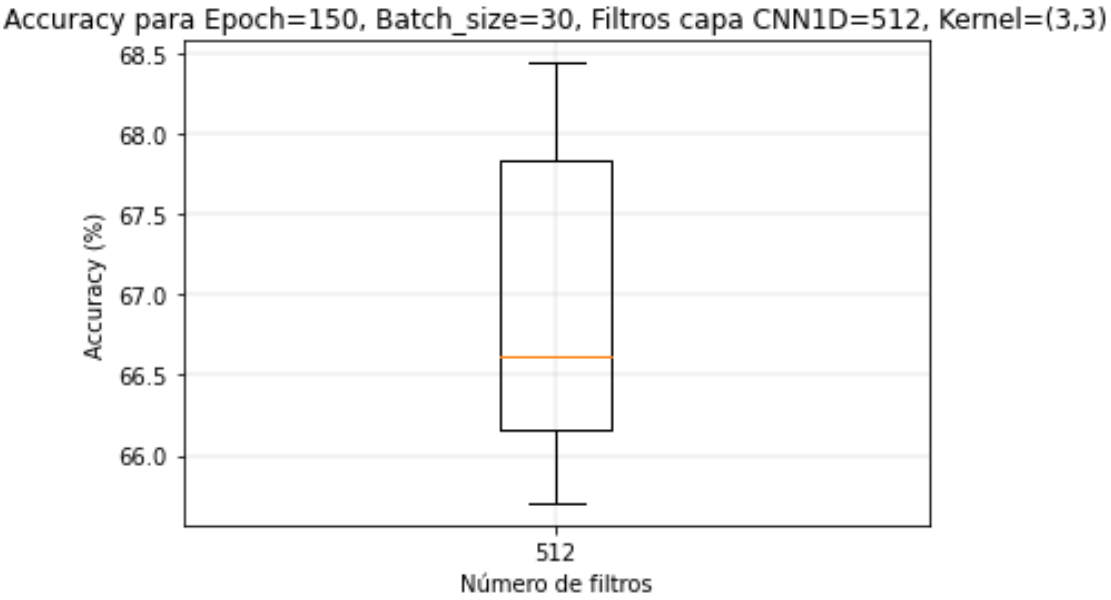
- A3.2.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



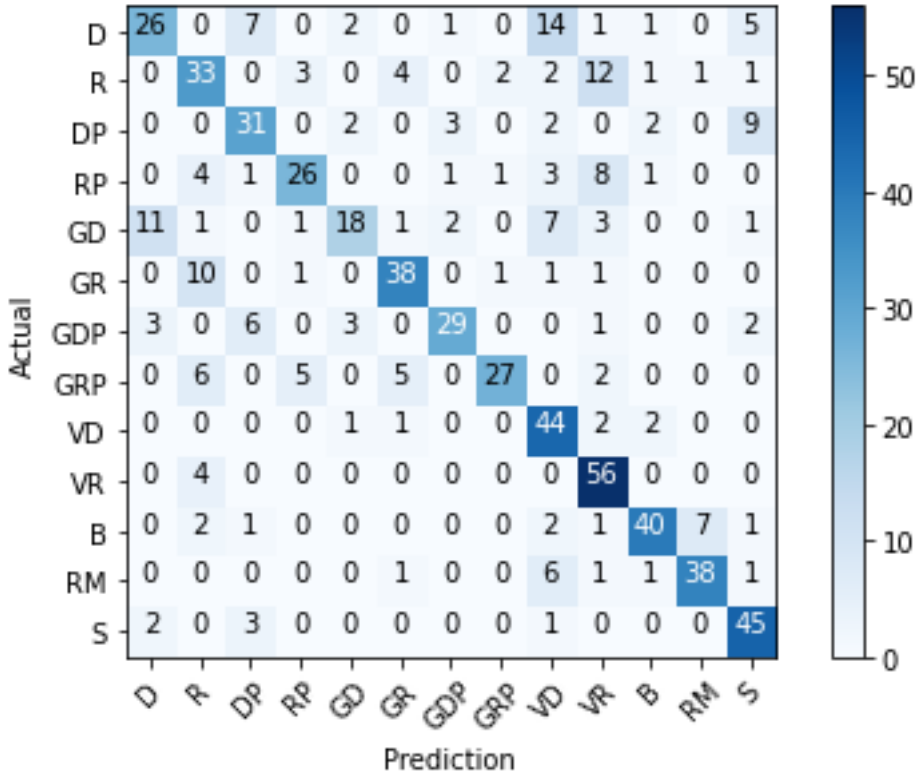
  - Matriz de confusión



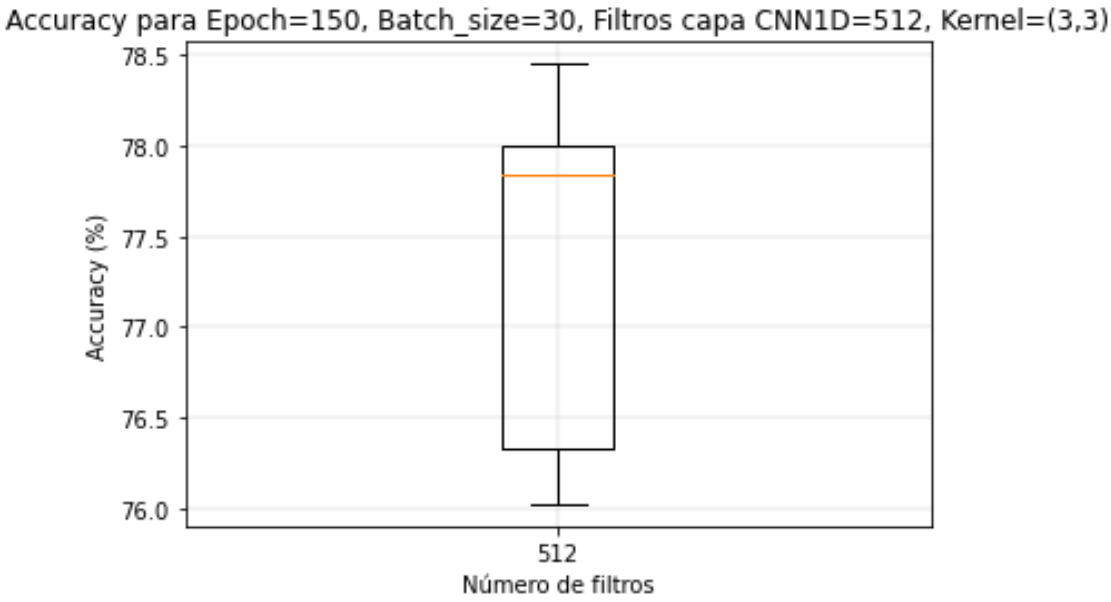
- A3.3 → REDES CONVOLUCIONALES 2D
  - A3.3.1 → Una capa convolucional y una capa densa
    - Diagrama de bigotes



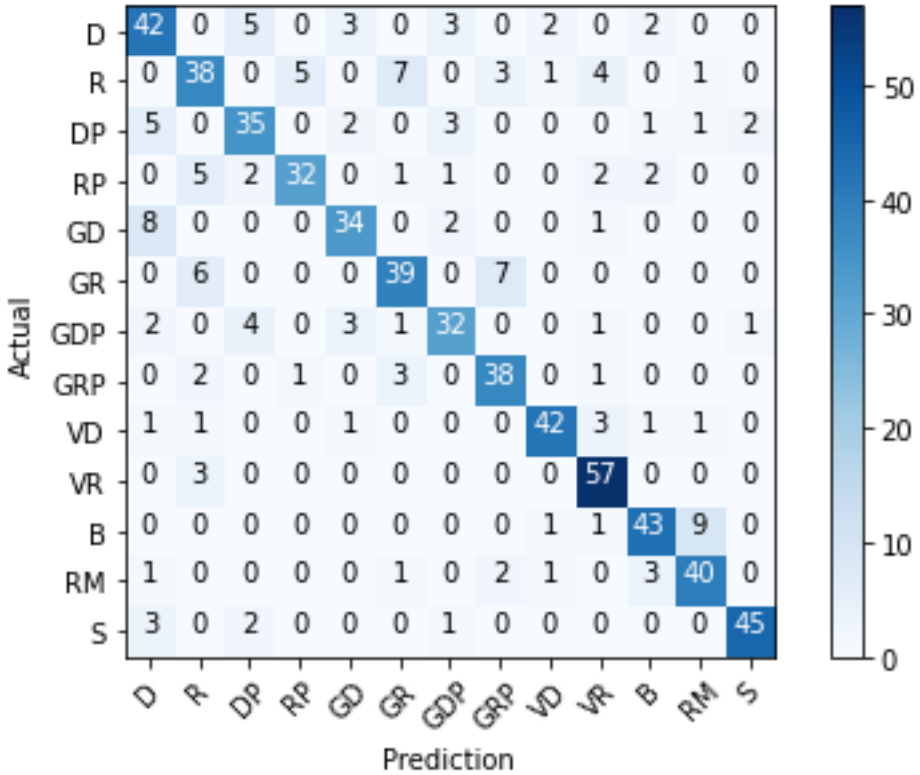
    - Matriz de confusión



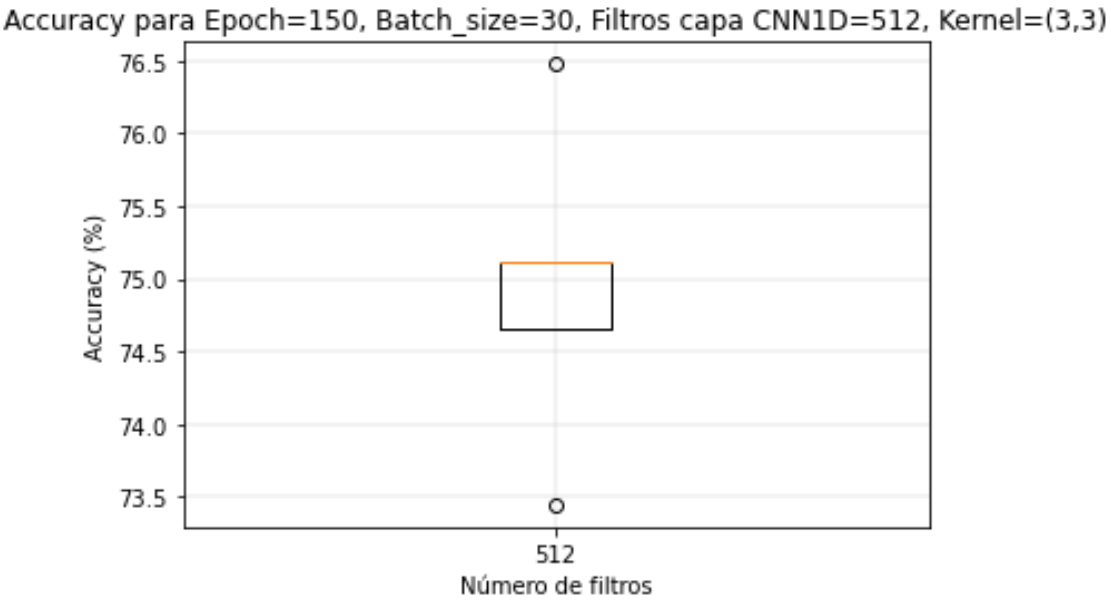
- A3.3.2 → Dos capas convolucionales y una capa dense
  - Diagrama de bigotes



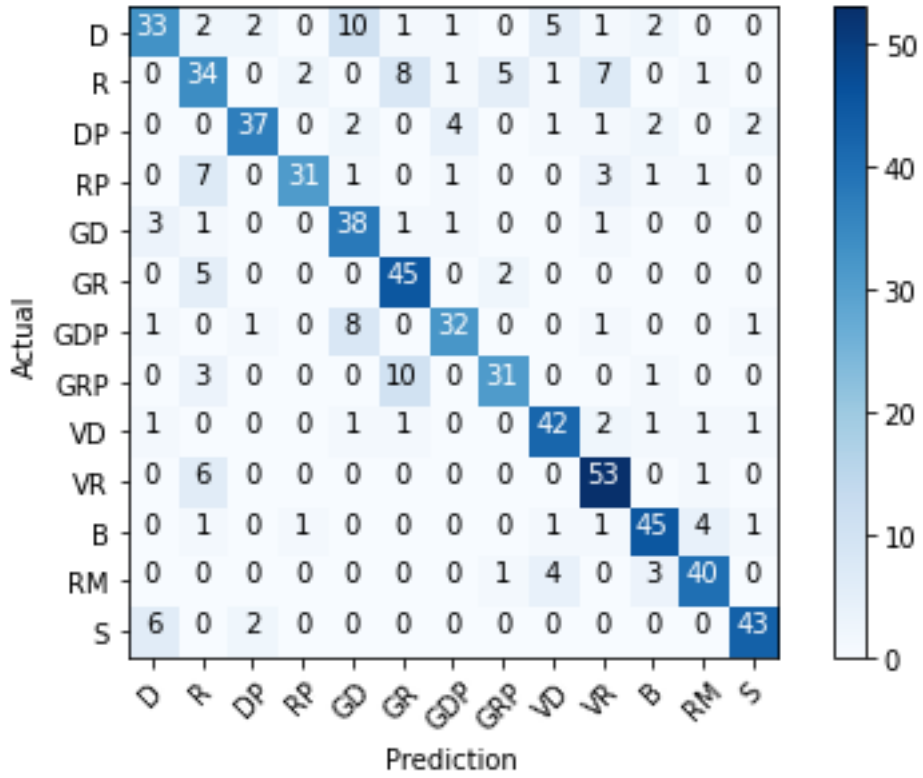
  - Matriz de confusión



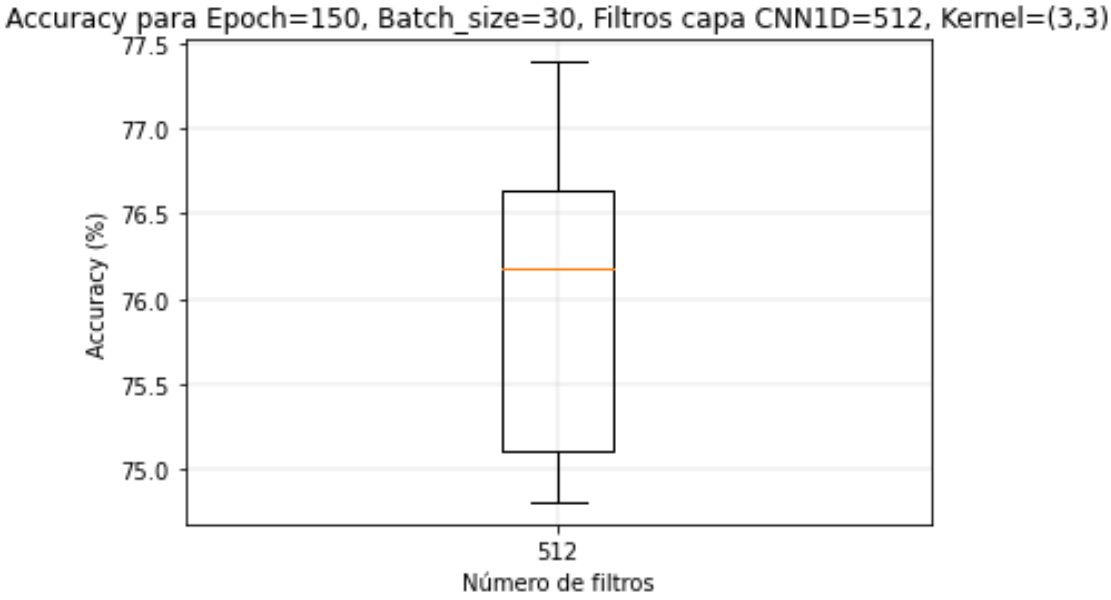
- A3.3.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



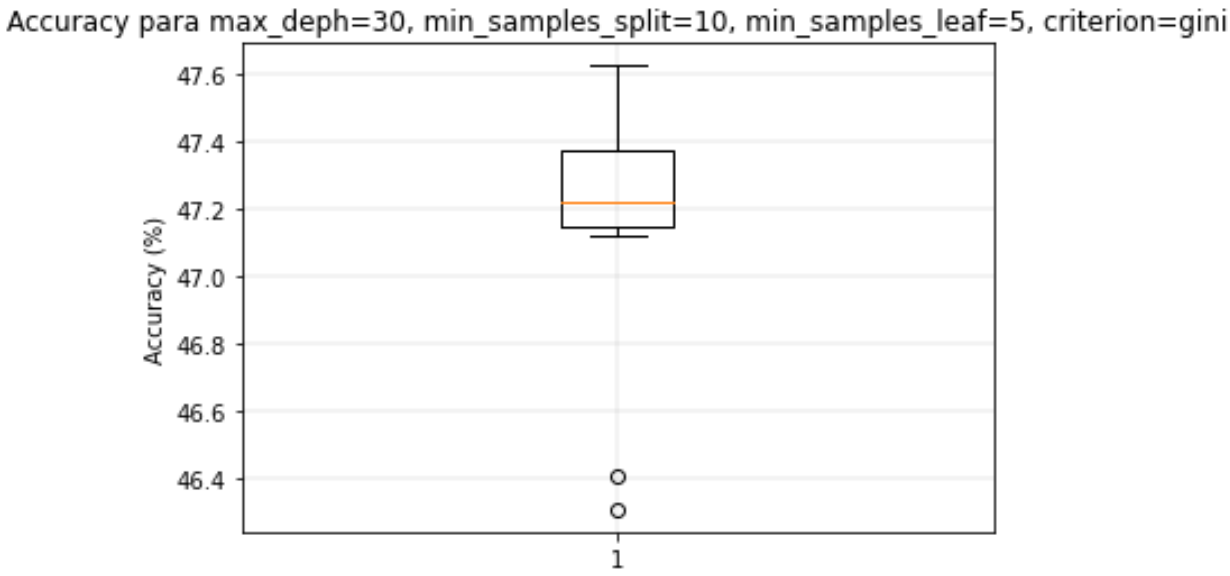
  - Matriz de confusión



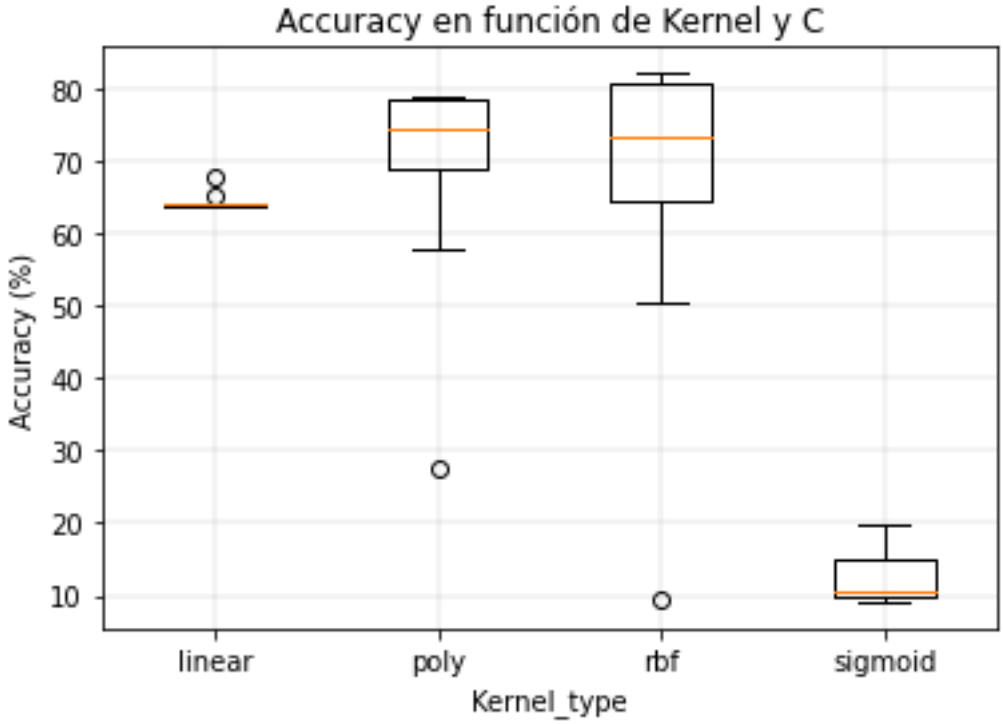
- A3.3.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



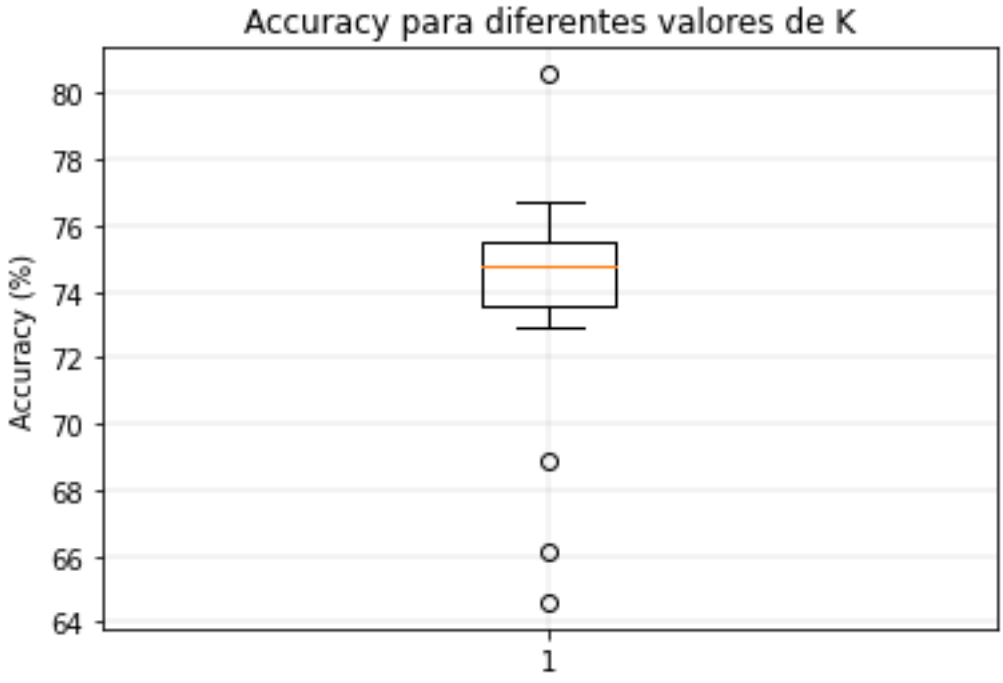
- A3.4 → ÁRBOL DE DECISIÓN
  - Diagrama de bigotes



- A3.5 → SVM
  - Diagrama de bigotes



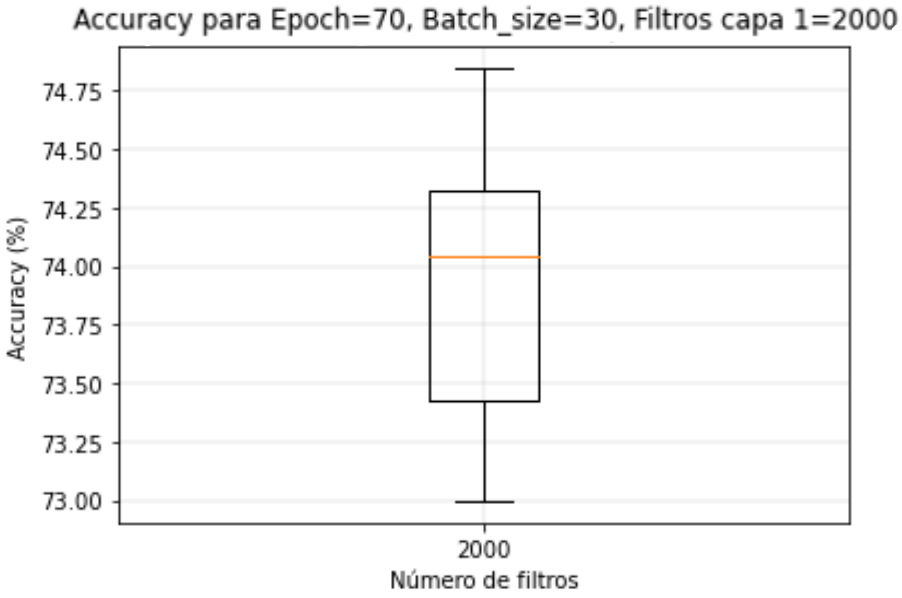
- A3.6 → KNN
  - Diagrama de bigotes



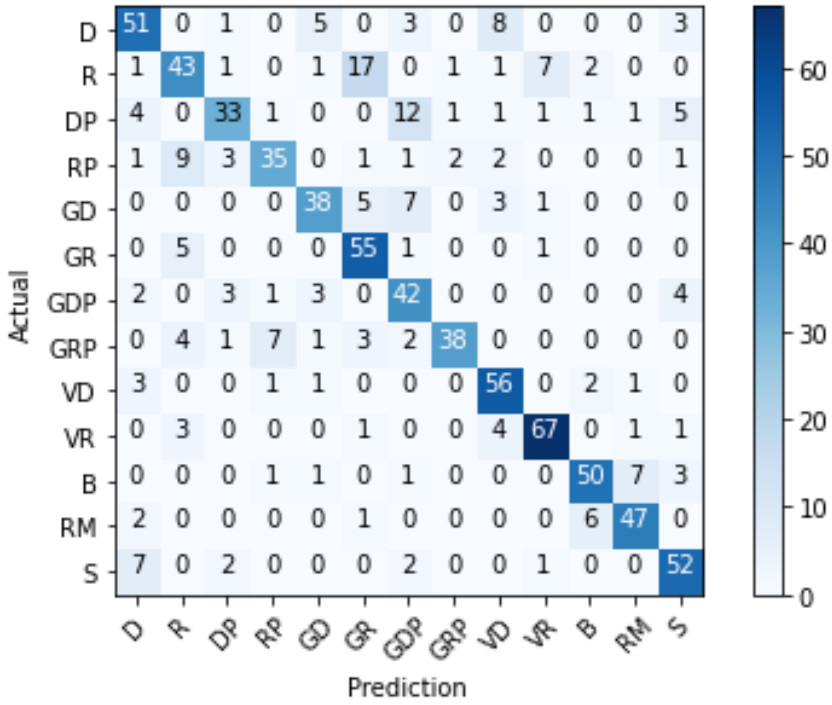
ANEXO IV

A lo largo de este anexo se mostrarán los diagramas de bigotes y las matrices de confusión de las mejores configuraciones para los distintos algoritmos para el caso de estudio en el que se entrena la red neuronal con la base de datos en el dominio de la frecuencia y con jugadores zurdos y diestros.

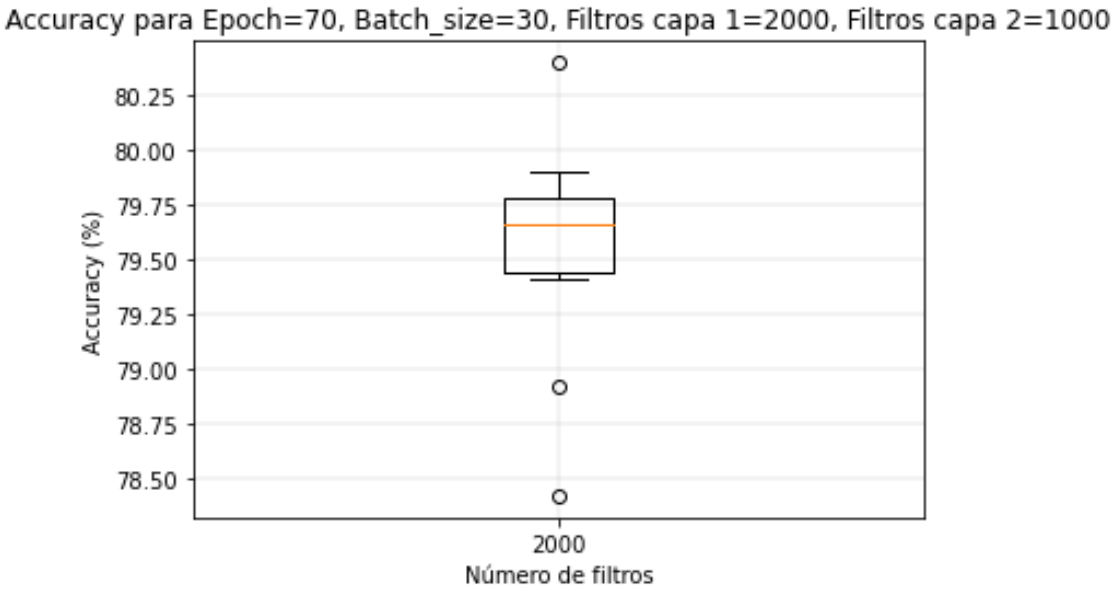
- A4.1 → REDES NEURONALES DENSAMENTE CONECTADAS
  - A4.1.1 → Dos capas densamente conectadas
    - Diagrama de bigotes



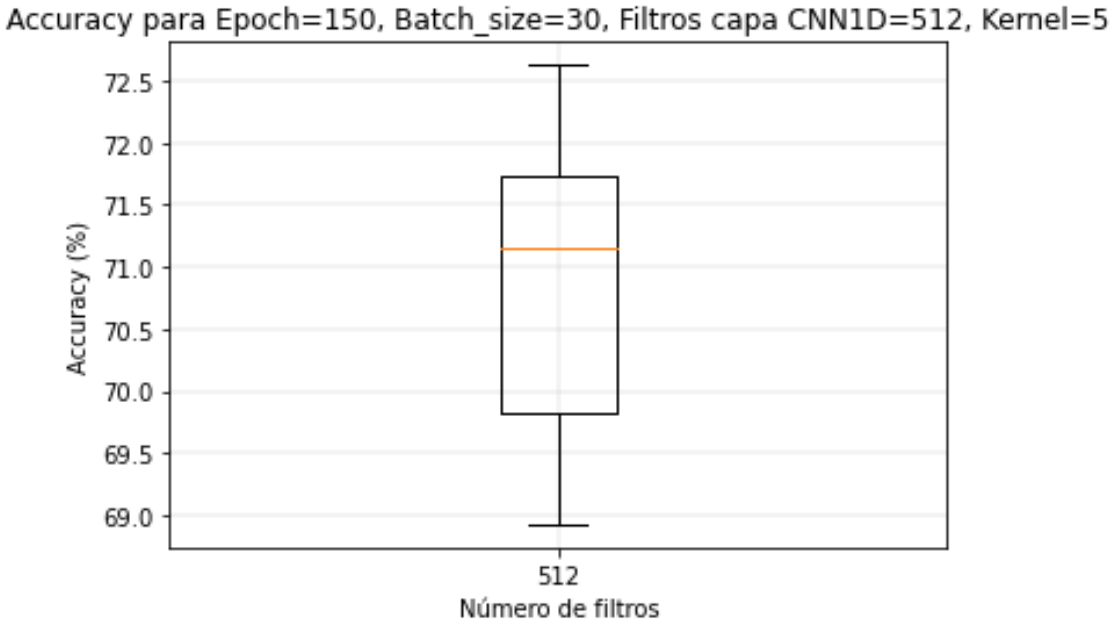
    - Matriz de confusión



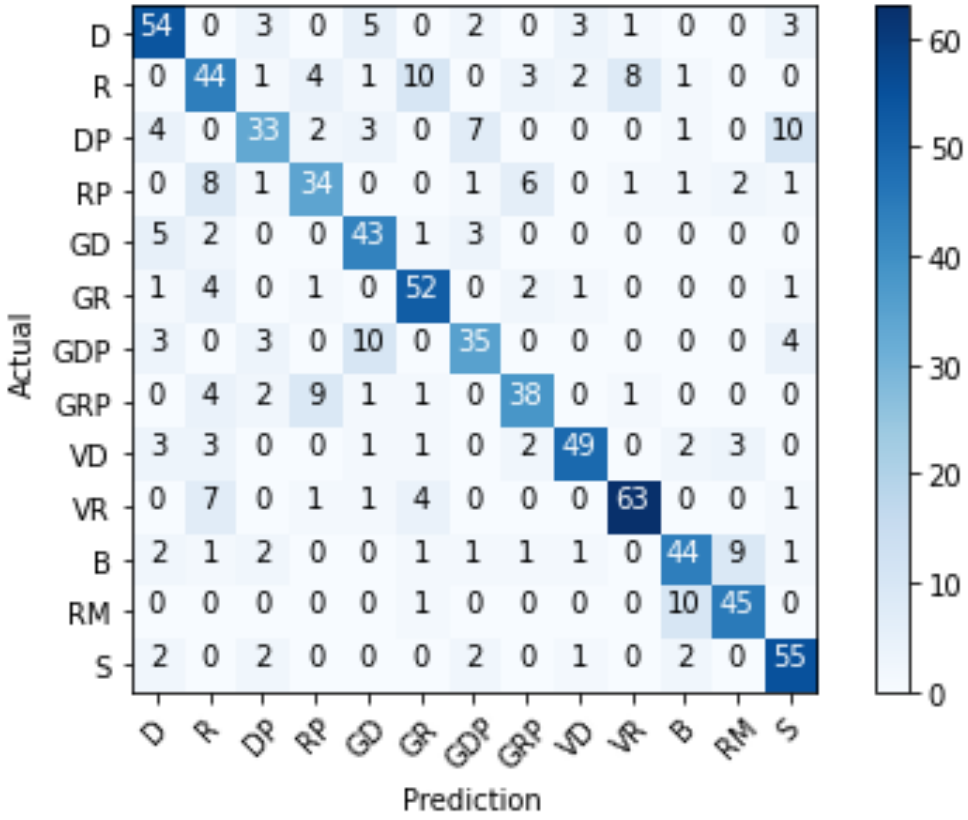
- A4.1.2 → Tres capas densamente conectadas
  - Diagrama de bigotes



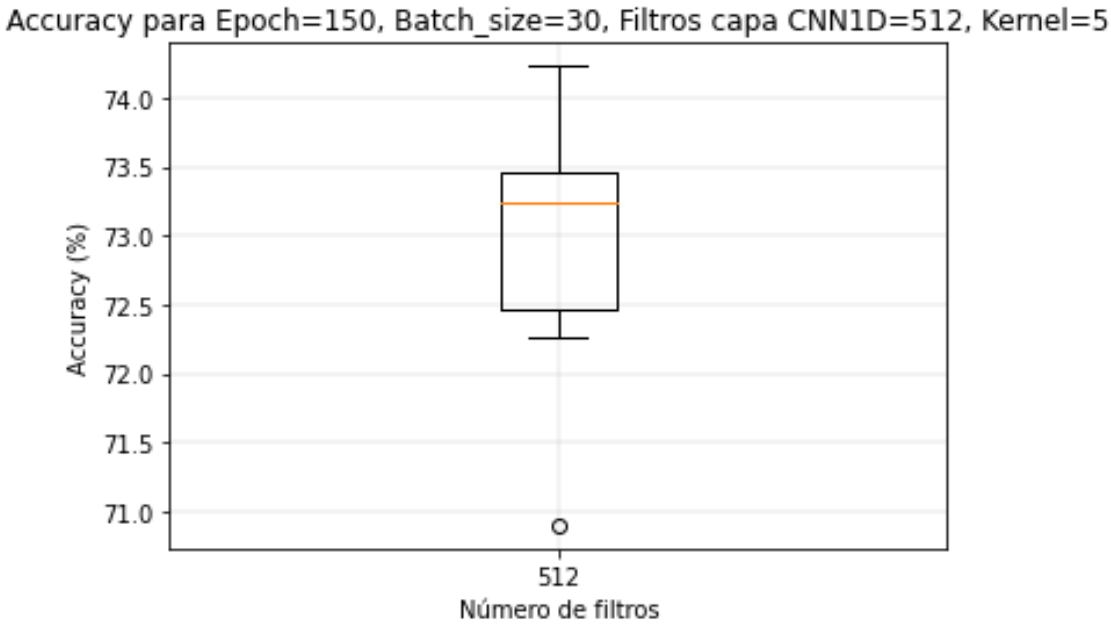
- A4.2 → REDES NEURONALES CONVOLUCIONALES 1D
  - A4.2.1 → Una capa convolucional y una capa densa
    - Diagrama de bigotes



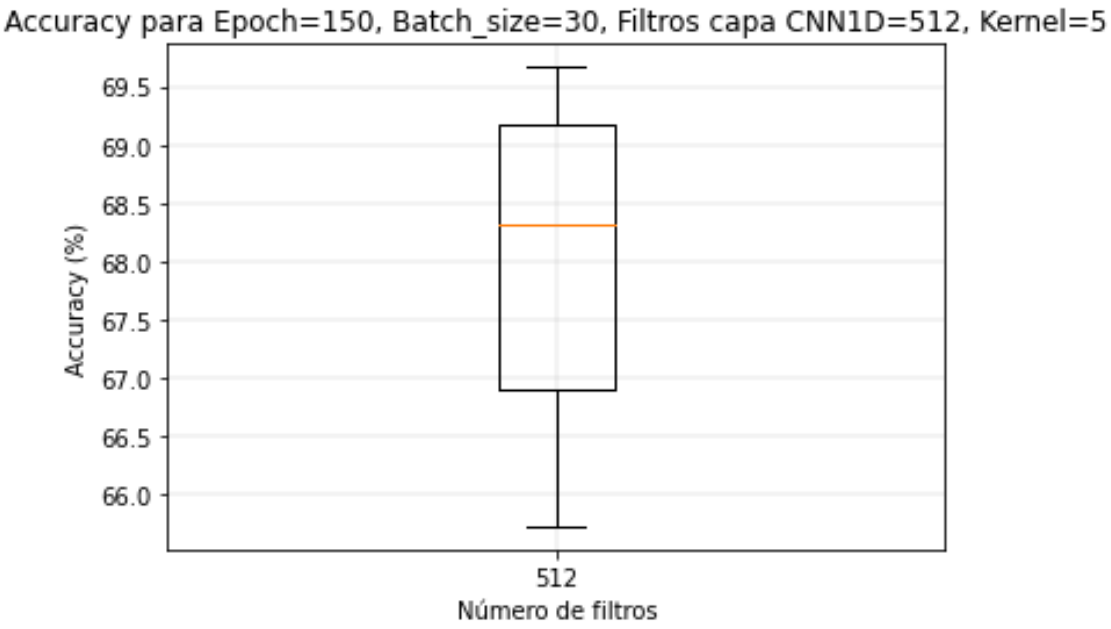
- Matriz de confusión



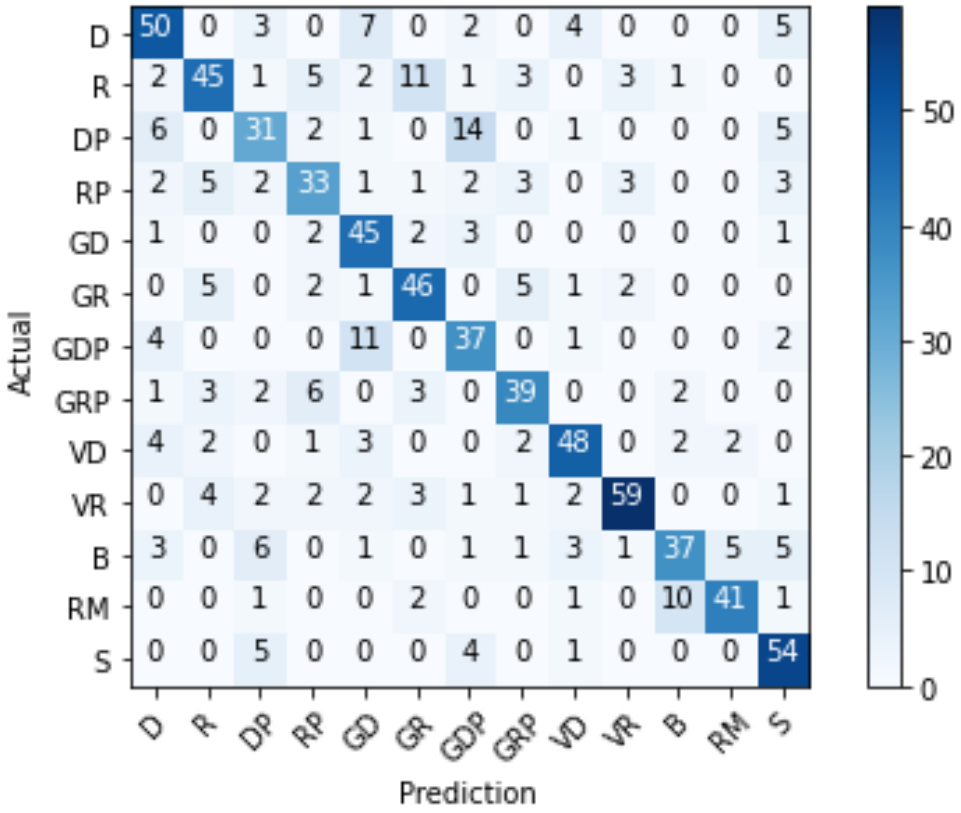
- A4.2.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



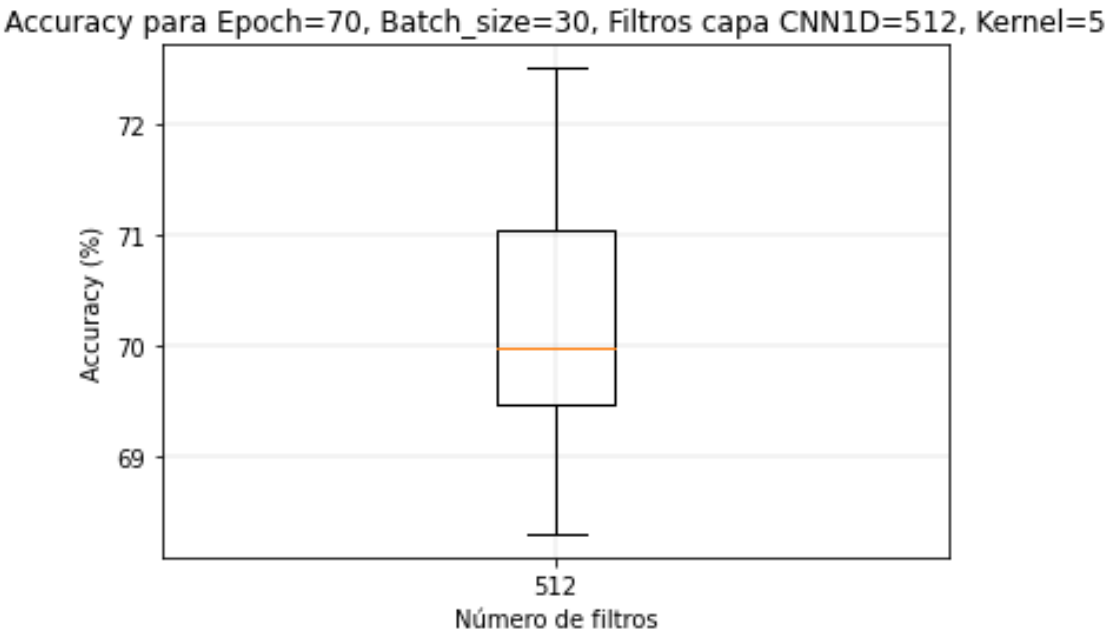
- A4.2.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



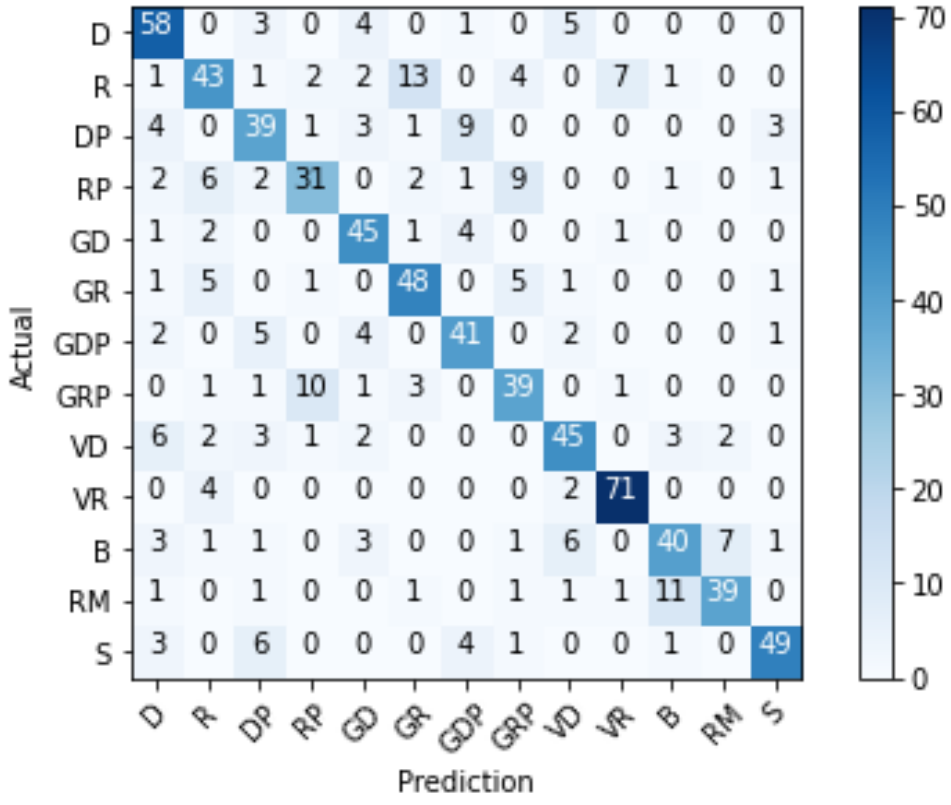
  - Matriz de confusión



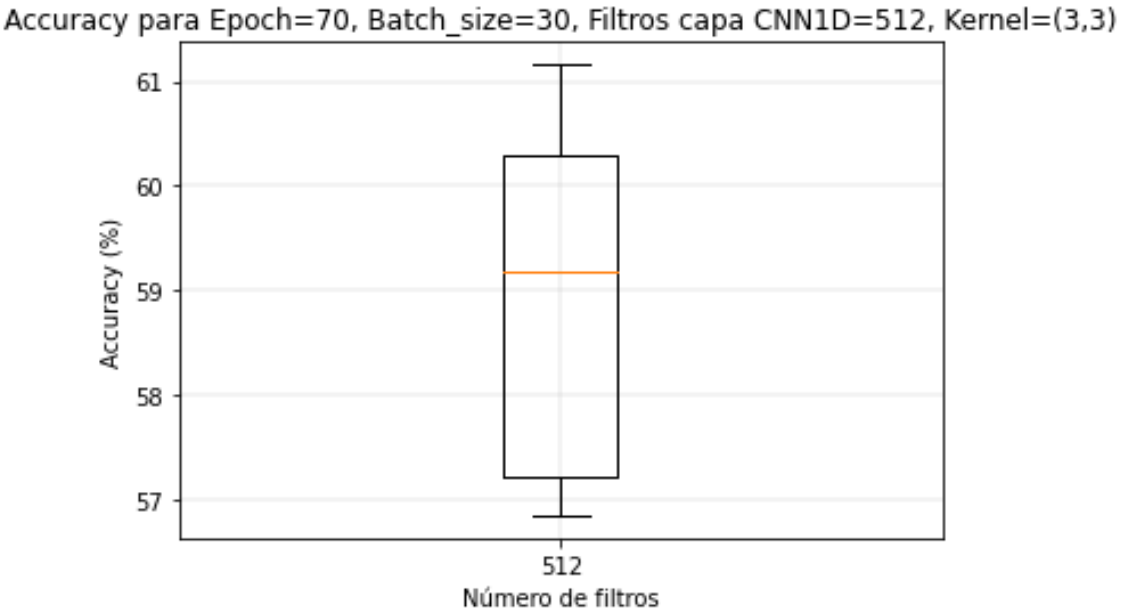
- A4.2.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



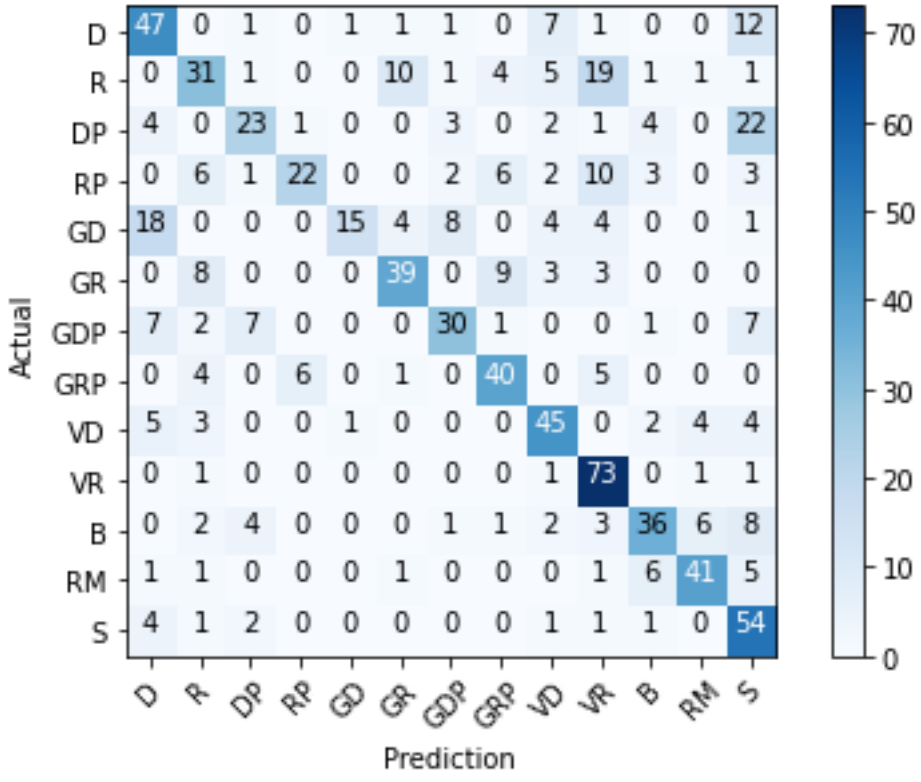
  - Matriz de confusión



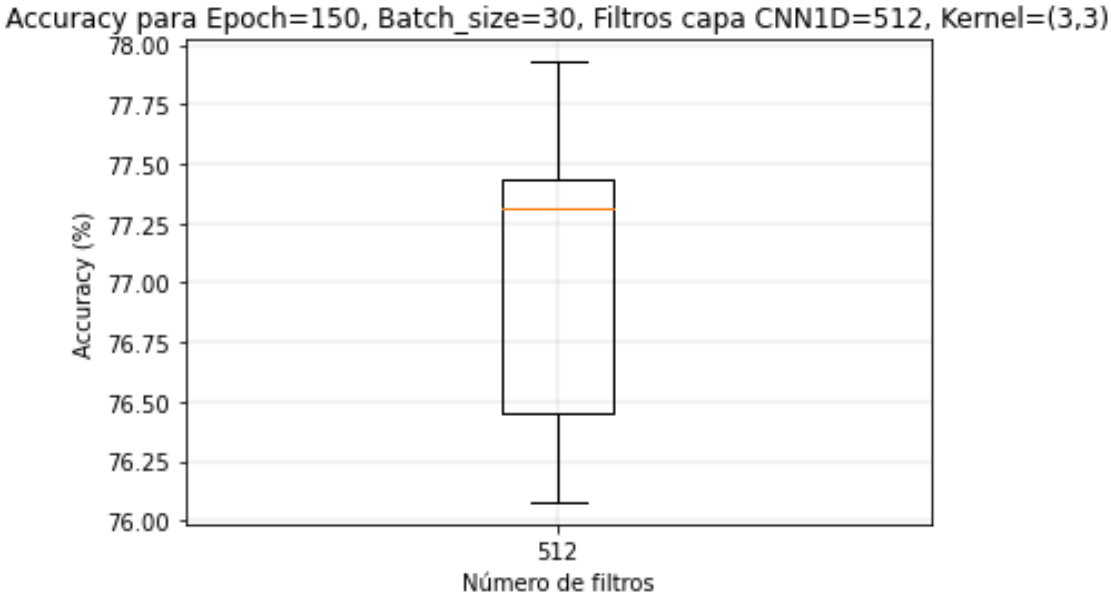
- A4.3 → REDES CONVOLUCIONALES 2D
  - A4.3.1 → Una capa convolucional y una capa densa
    - Diagrama de bigotes



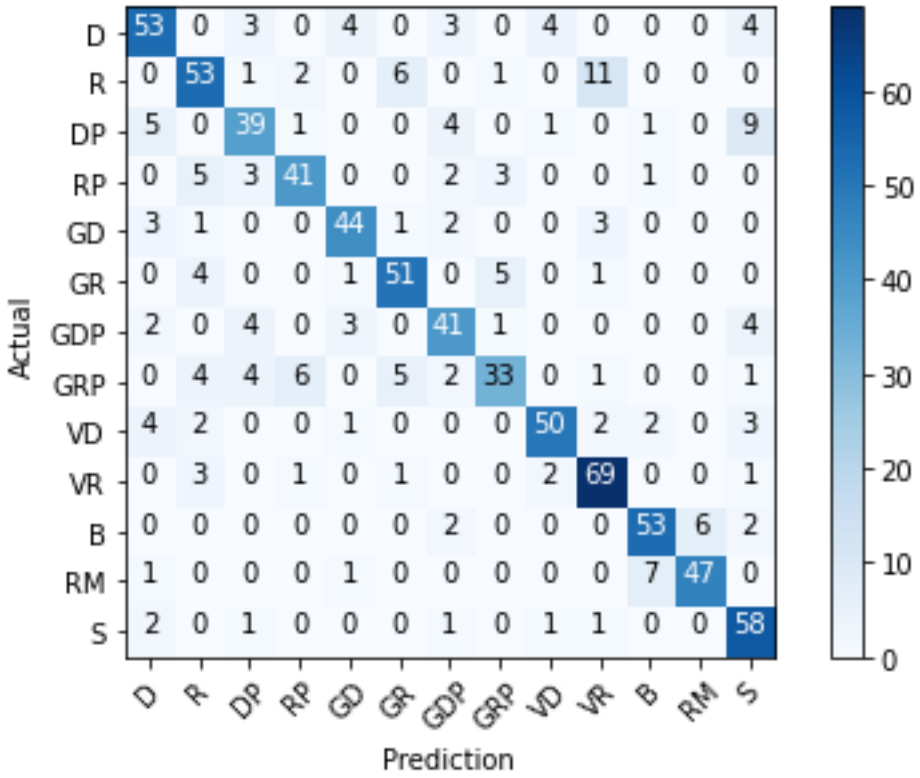
- Matriz de confusión



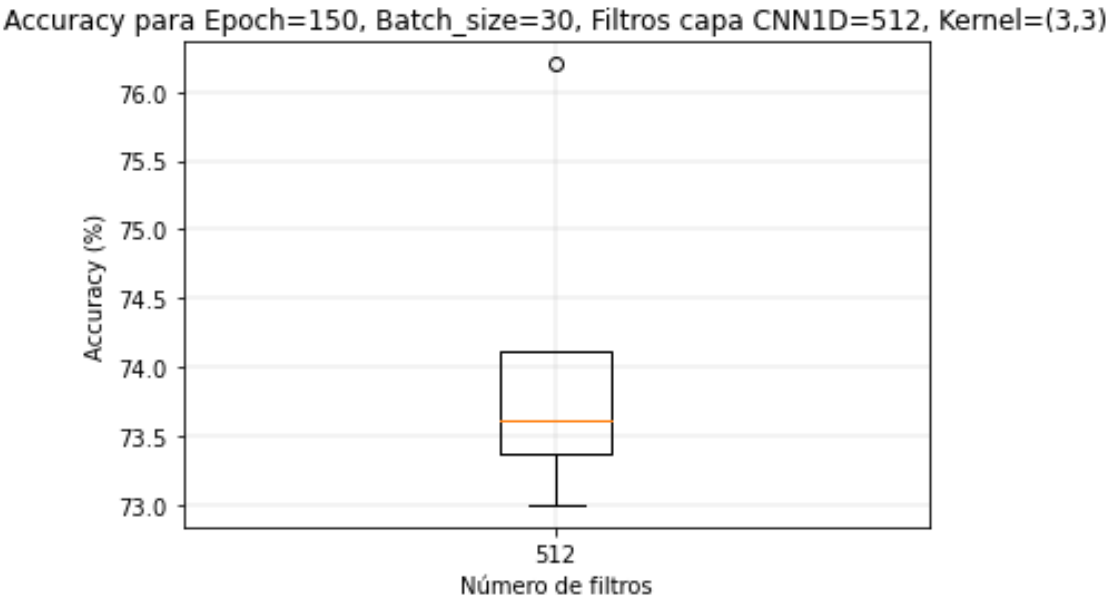
- A4.3.2 → Dos capas convolucionales y una capa densa
  - Diagrama de bigotes



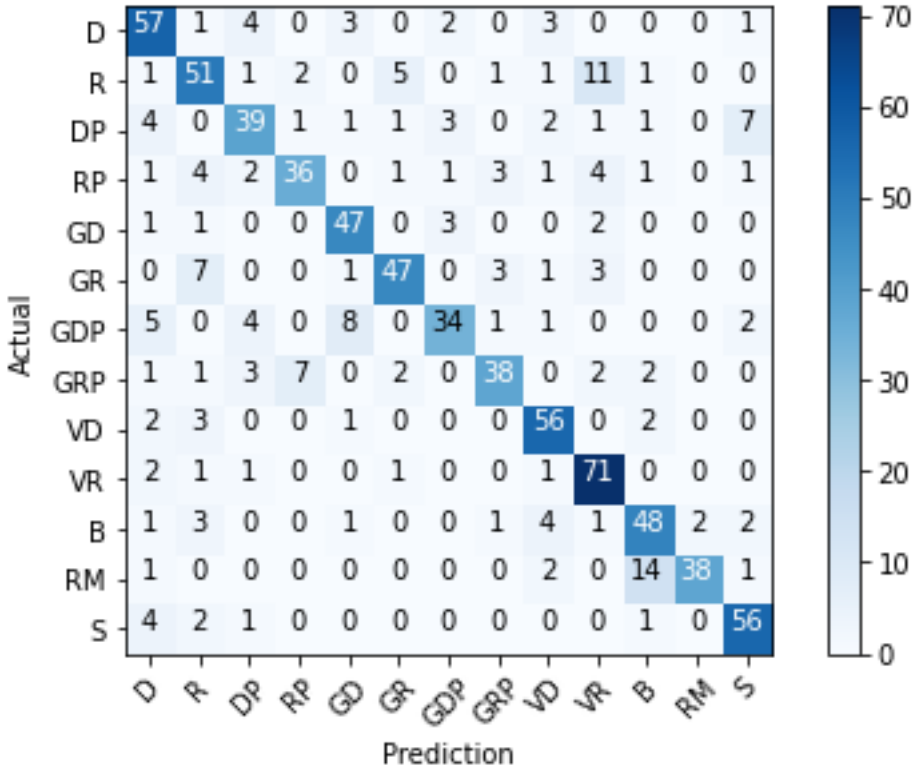
  - Matriz de confusión



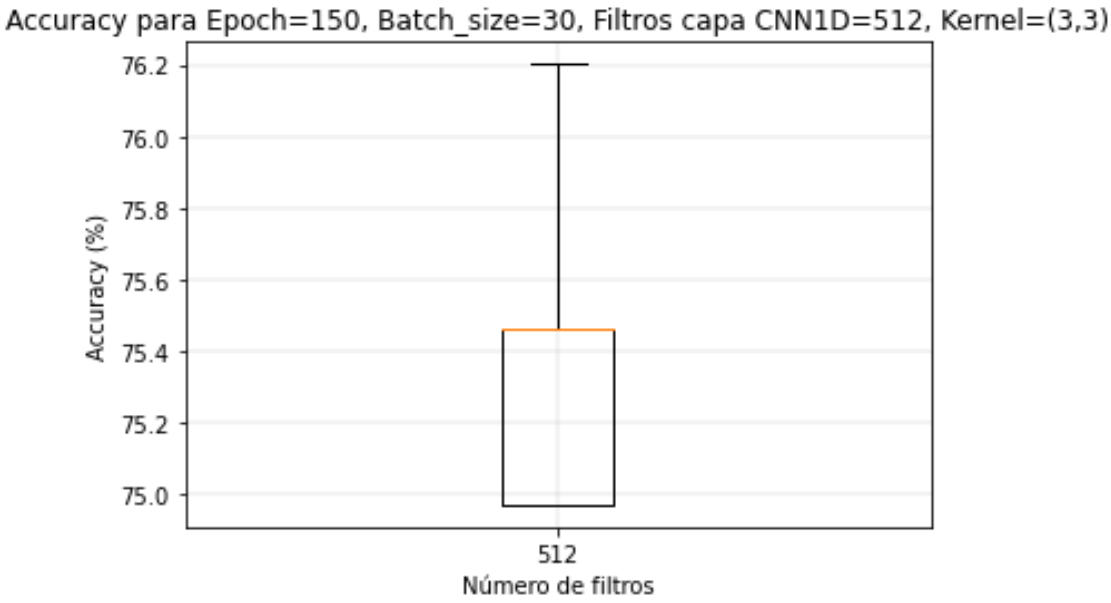
- A4.3.3 → Una capa convolucional y dos capas densas
  - Diagrama de bigotes



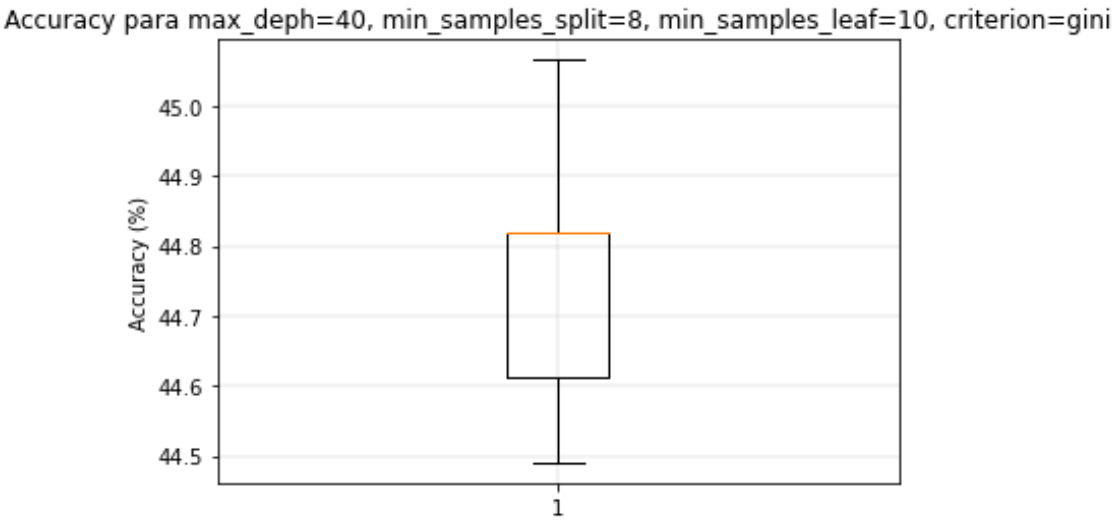
  - Matriz de confusión



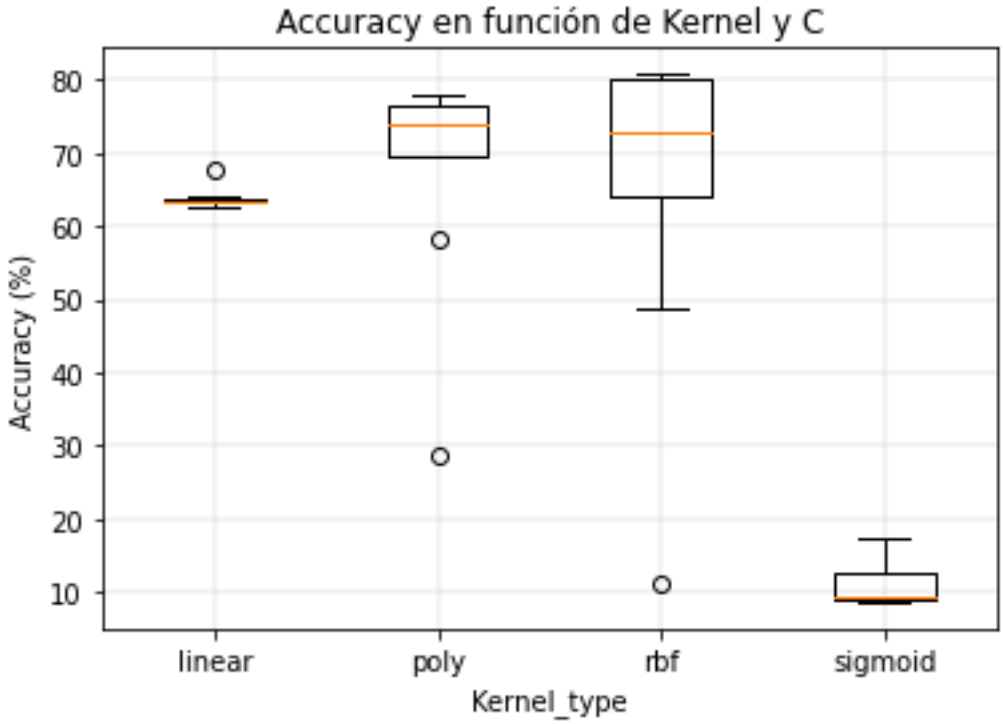
- A4.3.4 → Dos capas convolucionales y dos capas densas
  - Diagrama de bigotes



- A4.4 → ÁRBOL DE DECISIÓN
  - Diagrama de bigotes



- A4.5 → SVM
  - Diagrama de bigotes



- A4.6 → KNN
  - Diagrama de bigotes