



A comparative study of machine learning and deep learning algorithms for padel tennis shot classification

Guillermo Cartes Domínguez¹ · Evelia Franco Álvarez² · Alejandro Tapia Córdoba³ · Daniel Gutiérrez Reina¹

Accepted: 22 January 2023
© The Author(s) 2023

Abstract

Data processing in sports is a phenomenon increasingly present at all levels, from professionals in search of tools to improve their performance to beginners motivated by the quantification of their physical activity. In this work, a comparison between some of the main machine learning and deep learning algorithms is carried out in order to classify padel tennis strokes. Up to 13 representative padel tennis strokes are classified. Before a classification of padel tennis strokes is performed, a sufficiently representative data set is needed that collects numerous examples of the performance of these strokes. Since there was no similar data set in the literature, we proceeded to the creation of such a data set, for which we developed a data collection system based on an electronic device with an inertial measurement unit. Using the developed data set, the machine learning and deep learning algorithms were hyperparameterized to compare their performance under the best possible configurations. The algorithms were fed with both the temporal series of the acceleration and speed of the six degrees of freedom and also with feature engineering input, consisting in calculating the mean, maximum, and minimum values for each axis. The algorithms evaluated are: fully connected or dense neural networks, 1D convolutional neural networks, decision tree, K nearest neighbors, support vector machines, and eigenvalue classification. According to the results achieved, the best algorithm is the 1D convolutional neural network with temporal series input that achieves an accuracy higher than 93%. However, other simpler algorithms such as dense networks and support vector machines achieve similar results.

Keywords Padel tennis · Machine learning · Deep learning · Shot classification · Comparative study

1 Introduction

Over the last few years, the use of microelectromechanical systems (MEMS) has experienced great expansion in many sectors. It originated with the silicon revolution, thanks to two important inventions in the late 1950s: (i) the integrated circuit chip and (ii) the metal oxide semiconductor field effect transistor (MOSFET), which led to the miniaturization of electronics, as predicted by Moore's law. The mass production of these systems, together with the popularity of their use in a multitude of smart devices, has led to a drastic drop in their price, thus facilitating their expansion in almost all areas, among which wearables play a relevant role (Kos and Kramberger 2017). These are small devices that are placed on the human body, such as bracelets, watches, or rings, and interact with it and other devices for different purposes. To put the popularity of these systems into magnitude, the MEMS market garnered a market value of US\$14.32 billion in 2021. The MEMS market is expected to register a compound annual growth rate (CAGR) of 18.01% by accu-

✉ Daniel Gutiérrez Reina
dgutierrezreina@us.es

Guillermo Cartes Domínguez
guicardom@alum.us.es

Evelia Franco Álvarez
efalvarez@comillas.edu

Alejandro Tapia Córdoba
atapia@uloyola.es

¹ Department of Electronic Engineering, University of Seville, Camino de los descubrimientos, s/n, 41092 Seville, Spain

² Education, Research Methods and Evaluation Department, Comillas Pontifical University, Calle Universidad de Comillas, 3-5, 48049 Madrid, Spain

³ Department of Engineering, Universidad Loyola Andalucía, Avenida de las Universidades, s/n, 41704 Dos Hermanas, Spain

mulating a market value of US\$ 75 billion during the forecast period 2022–2032 (FACTMR, 2021).

Given the huge amount of data that these devices can easily provide, wearables are becoming popular for sports practice at all levels, from fans at home to professional athletes (Seshadri et al. 2019; Yang et al. 2021), drawing the attention of coaching staff, doctors, and clubs. This technology makes it possible to perform a variety of actions such as evaluating player performance or compiling and comparing match statistics. It is here where big data has paved the road for the development of artificial intelligence (AI), since the appropriate management of large amounts of data allows an effective learning from it, enabling machines to make decisions that can be qualified as intelligent.

Any data generated on the playing field of play can be analyzed and studied. For example, in a tennis match lasting an hour and a half, around 70.000 records are generated, while in a soccer match, this number may be increase up to eight million pieces of captured data. It is clear that this enormous amount of information can only be analyzed using data-based technologies. Although the pioneer in the use of big data was baseball in the 1970s, when historical records of baseball players in the major American leagues began to be used (Ladany and Machol 1977), there are numerous applications that have subsequently arisen through adaptations to different contexts. In the case of basketball, for example, NBA data have shown that shorter and more versatile players are more efficient than taller players who defend the rim, as it is more productive to score three-pointers than two-point baskets, even assuming worse defense on the rim itself (Ladany and Machol 1977). In soccer, big data brings great value to all clubs in the world, as they use this technology to improve their game strategies, sign new players, etc. Data processing in sport can also be applied for other purposes, for instance, the prevention of injuries. A recent study estimated that clubs lose between 10% and 30% of their squads due to injuries (Päivitetty, 2015). With the use of the appropriate technology, any deficiencies in an athlete's fitness can be early identified and certain injuries can be prevented.

Despite its recent creation (Martínez 2013), padel tennis is a sport that is booming worldwide. For instance, in Spain, padel tennis has doubled the number of federated players in less than 9 years, surpassing tennis in 2020, and becoming one of the most practiced sports at national level.¹ Therefore, this study aims to bring together two growing disciplines: artificial intelligence and tennis padel. This paper presents a comparison of deep learning and machine learning algorithms for the classification of padel strokes. A stroke is given when a player hits the ball with the racket. Then, the target



Fig. 1 Electronic wearable device placed at the wrist of the player. It is based on Raspberry pi 4 with Sense HAT module that incorporates an IMU sensor

problem consists of identifying the type of hit given by a player. For this purpose, temporal data collected by an electronic wearable was used. This wearable was placed on the wrist of the player (see Fig. 1). The data gathered came from an inertial measurement unit (IMU) and included both angular acceleration and velocity. Since the variety of strokes in the padel is high (even higher than in tennis), including smash, tray, lob, etc.², it leads to a complex multiclass classification problem. The data input consisted of temporal data of the angular acceleration and speed during a temporal window that includes the preparation of the stroke, the stroke itself, and the continuation of the stroke. Notice that these three aspects are important to identify the type of stroke given by a player.

For the target classification problem, we propose to evaluate the performance of data-based algorithms to take advantage of the data generated during a stroke. The algorithms evaluated range from classical machine learning algorithms, such as decision trees (DT) and support vector machines (SVM) Zhou and Gan (2021), to neural networks approaches (Voulodimos et al. 2018) like dense networks and 1D convolutional neural networks (CNN) (Ragab et al. 2020; Gutiérrez and Toral 2019), including distance or dissimilarity-based algorithms such as K-means and Eigenvalue algorithm (Erkan 2021). To train the algorithms considered, data related to padel strokes are required. However, no structured data related to the padel strokes is available in the literature. Therefore, a specific data set was created for the purpose of training the algorithms. Thus, and to the best authors' knowledge, this is the first data set about padel in the literature. It includes mixed-gender players of several levels and more than 2000 strokes of up to 13 different types.

In summary, the main contributions of this study are as follows:

- Elaboration of the first data set of padel tennis strokes. Data of more than 2000 strokes have been recorded and structured, including players of different levels and genre.
- A comparison of deep learning and machine learning algorithms for stroke classification. Up to 13 different

¹ Data of sport practitioners in Spain are available in <https://www.csd.gob.es/es/federaciones-y-asociaciones/federaciones-deportivas-espanolas/licencias>.

² <https://sportsclubtour.com/the-top-10-padel-shots/>.

types of strokes are classified by the algorithms. For the sake of an unbiased comparison, a thorough hyperparameter analysis has been conducted for each algorithm.

- A comparison between temporal series input versus, consisting of 40 timestamps for 6 features (acceleration and speed for each axis, x, y, and z); versus feature engineering, consisting of an array of 18 components (mean, maximum, and minimum for each degree of freedom).

The rest of the paper continues as follows, Sect. 2 presents some similar works proposed for racket sports. Section 3 describes the creation of the data set. Section 4 presents the machine and deep learning algorithms used for the comparison, detailing the main hyperparameters that affect their performance. Performance evaluation comparison can be found in Sect. 5 that includes: the hyperparameterization of the algorithms and their comparison under the best possible configuration. The paper ends with Sect. 6 which contains the conclusions of this work and future research directions.

2 Related work

Stroke classification has been extensively used for other racket sports, such as tennis and table tennis (McGrath et al. 2021). For example, the authors in Kos et al. (2016) presented a study on tennis stroke detection and classification. This was conducted with a sensor with 6 degrees of freedom (DOF) and a sampling rate of 1000 Hz that was placed on the wrist of the tennis player. Data collection was performed with 3 different people with playing skills, providing a total of 147 strokes of three types: serve, forehand, and backhand. For stroke detection, they used an experimentally established threshold on the gyroscope derivative: When this threshold was exceeded, a stroke was considered to have occurred. Regarding classification, they used a simple algorithm, built on the basis of previous observations of the differences between the three strokes studied. This algorithm provided a 98.1% accuracy. In Whiteside et al. (2017), the authors addressed the same problem but employed a machine learning approach. They used data from 19 different athletes (8 women and 11 men) to classify 9 types of stroke. They collected 28,582 strokes using an IMU placed on the athlete's wrist, sampling at a rate of 500 Hz. They studied the behavior of six machine learning models, achieving an accuracy of 93.2%. A similar approach is proposed in (Benages Pardo et al. 2019), where the authors conducted a study on activity detection with two sensors, one placed on the wrist and the other on the waist of the athlete, providing a total of 12 DOF (6+6). As for the strokes, four different stroke types were considered: forehand, backhand, lob, and volley. Data were collected from 8 persons, 4 men and 4 women, of whom 7 were right-handed and 1 left-handed. The results of the stroke classification were 99.25% correct

for training and 96.51% correct for test. In Wu et al. (2022), data from both the accelerometer and the gyroscope are used to classify five strokes in tennis. The authors used a data set composed of 36 participants ranging from different levels (12 elite, 12 subelite, and 12 amateur). The wearable device was placed on the wrist of the players, and the sample frequency was 50 Hz. The authors used the SVM algorithm for the classification task and compared the results with the KNN algorithm. Although tennis constitutes the most extended application of stroke analysis in the literature, table tennis has also been extensively studied using similar approaches. For example, the work in Tabrizi et al. (2020) presents a comparative study of table tennis strokes using deep learning. They collected data from 16 participants divided into two groups: beginners and professionals. A total of 1080 strokes were taken from the first group, while 648 strokes were taken from the latter, without any additional considerations, such as gender, age, or height. In Sha et al. (2021), the authors use several machine learning algorithms to identify table tennis players using data collected from players' wrist during shots. They propose to use temporal and frequency features, which are obtained from the raw data. In addition, they use Laplacian scores to select the most suitable features for training the machine learning models. The developed models achieved a 99% of accuracy in terms of player identification, being SVM model the one that obtains the best performance.

Padel tennis still has a shorter track record in the literature, especially with respect to data analysis, but some studies are worth mentioning: An analysis of the distance traveled in padel tennis is carried out as a function of the level of play and the number of points per match. For its development, two overhead video cameras were employed. These were placed on top of the roof of the court, 9 ms high from the ground, each focusing on a part of the court. Samples were collected from 108 federated players during 27 matches, for a total of 4,406 points. The results showed that a player travels an average of 11 ms per point and 2,900 ms per match, divided into 51% in the active phase (playing time) and 49% in the passive phase (rest time). Players were classified into 3 levels, and it was concluded that midlevel players covered almost 400 ms more in the active phase than high-level players and almost 900 ms more than low-level players. In Ramón-Llín et al. (2021) an analysis of the final strokes of the point is presented in padel tennis using a decision tree. A total of 2,110 game actions are collected and analyzed: stroke, area of the court, efficiency, direction, result, and side of play. In this case, a video camera was used placed 1.5 ms high and 3 ms behind the court. A total of 1,055 points were collected from 9 top-level padel tennis matches, with 36 different players. The results showed that holding positions close to the net increases the probability of winning, with the most frequent finishing sequences being backhand-volley and lob-rebound. Furthermore, using

crossed trajectories on the penultimate shot increase the chances of a subsequent error by the opponents.

The present work fills several gaps in the current literature. First, it presents the first data set available on padel tennis strokes. It will allow other researchers in the area to develop new approaches based on the collected data. Second, none of the previous works based on racket sports has addressed a classification problem with up to 13 different classes. Previous work related to stroke classification in tennis (Whiteside et al. 2017; Kos et al. 2016; Wu et al. 2022) only considered up to 9 different shots. In addition, in padel tennis, some shots have little differences in terms of execution and hand movement, since the main difference is that the wall is used before hitting the ball. Therefore, this work presents a complex classification that has not been solved to date. Third, the 1D convolutional neural architecture used in this work has not been proposed in any of the previous works. Since this approach achieves the best results, these results can be transferred to other racket sports.

3 Data set creation

The first challenge with regard to the classification of padel tennis strokes lies in the generation of a representative data set from experimental data. As the data set is to be used for training and testing supervised learning algorithms, the different shots are required to be properly labeled. A set of experiments were carefully designed aiming at an efficient generation, labeling, and trimming of the experimental data series, in accordance with the different strokes performed. The procedure has been divided into four different stages, covering (i) the organization of the players and features involved, (ii) the acquisition of the raw data series, (iii) the shot detection and segmentation, and (iv) target labeling.

3.1 Organization of the data set

A wide range of different racket strokes can be identified in padel tennis, especially those in which spin is applied to the ball or in which the hit is performed after the ball bounces off the wall. For this work, the advice of two professional coaches was taken into account to select the most common strokes in the sport resulting in the set of 13 different shots which were finally analyzed. These included the distinction between strokes performed before and after the wall bounce (labeled as ground/wall strokes). Thus, the strokes proposed were: forehand ground, backhand ground, forehand wall, backhand wall, forehand lob, backhand lob, forehand lob wall, backhand lob wall, forehand volley, backhand volley, tray, smash, and Serve (see Sect. 3.4 for more details).

With respect to the participants responsible for the execution of the different strokes, a total of 12 different people

were involved in the data set creation. These people were a representative sample of padel tennis players in terms of gender, level, and age, among other aspects. A summary of the different characteristics of the participants is shown in Table 1. Some of these characteristics, such as age, sex, or height, do not require further explanation. The rest of them are explained below:

- The player's ID is a unique identifier for each subject.
- The level refers to the skill of the player. Given the complexity of its evaluation, the expert criteria of two professional coaches were considered. The coaches' advice was especially relevant to distinguish intermediate-level and high-level players. The following classification system was proposed:
 1. Beginner-novice (level 1): Casual player with less than a year of experience.
 2. Amateur (level 2): Casual player with more than one year of nonprofessional experience.
 3. Intermediate-level experience (level 3): Semi-professional player with experience in local or regional competitions, considered intermediate player at least by two national trainers.
 4. High-level experience (level 4): Semi-professional player with experience in local or regional competitions, considered an advanced player at least by two national trainers.
 5. Professional (level 5): Professional player with experience in the World Padel Tour (the most important international padel tennis competition).
- The backhand indicates whether the player performs the backhand using a single hand (1) or a two-hand (2) grip. It is relevant to note that, in padel tennis, backhand strokes are commonly performed with a single hand grip. However, the two-hand grip typically used by former tennis players.

Regarding the limitations of the collected data:

- The data does not include any left-handed players.
- Different effects of the strokes were not considered in the classification. However, when collecting data, specific instructions were given to the participants to hit the ball as flat as possible.
- The final data set was not revised to guarantee that participants were performing the strokes they had been instructed to do. This limitation could have mainly affected the data collected from novice players, which might have had some issues in distinguishing similar strokes (for instance, a tray and a smash). Note that it can affect the performance of the classification of some similar strokes.

Table 1 Summary of players characteristics

ID	Sex	Level	Hand	Backhand	Height (cm)	Age	Strokes performed
1	M	1	R	1	178	37	109
2	F	2	R	1	157	34	157
3	M	2	R	1	187	23	136
4	M	3	R	1	175	47	313
5	M	4	R	1	176	38	269
6	F	5	R	1	170	28	231
7	M	2	R	1	173	50	183
8	F	1	R	1	164	23	131
9	M	3	R	1	185	30	181
10	F	3	R	2	167	25	236
11	M	4	R	1	184	32	163
12	F	5	R	1	164	22	219
Total							2328

3.2 Raw data acquisition

To generate the data, an IMU was used, responsible for measuring 3 linear accelerations and 3 angular velocities at a 20 Hz rate. In particular, the IMU was an LSM9DS1, which is incorporated into the Sense Hat of the Raspberry Pi. Both the Raspberry Pi and the Sense Hat were fixed to the wrist using a 3D printed ergonomic platform, which in turn was firmly attached to the wrist with a Velcro-fixed band.

During capture routines, the Raspberry Pi was powered via USB using a portable external battery that was attached to the athlete's waist. Finally, the Raspberry Pi was connected, thanks to VNC Viewer, to a laptop wirelessly via a Wi-Fi network provided by a smartphone.

To facilitate the process, the data capture routines have been carefully designed to simplify the postlabeling labor. To this end, the process was divided into 13 different tests, one for each type of stroke considered. During each test, the subject performed multiple executions of a single type of stroke, for which counted on a second player, responsible for throwing the ball. After each of these tests were performed, a total of 13 sets of temporal series (each set consisting of data from the 6 DOF) were generated. With this structure (see Fig. 2), each set of temporal series corresponds to a compilation of numerous executions of a single type of stroke, which strongly simplifies the labeling.

3.3 Shot detection and segmentation

Once the temporal series had been created, individual strokes were required to be properly identified, trimmed, and labeled. To this end, a first visual inspection of the data showed that the different strokes exhibited a distinctive pattern at the moment of the ball hitting, with sudden and significant variations in almost all DOFs, reaching extreme local values in both angu-

lar and linear accelerations. On the basis of this evidence, a threshold value was defined for each DOF (u_a for linear accelerations and u_v for angular velocities), so that, if this threshold is reached simultaneously by any of the accelerations and any of the velocities, it can be assumed that a stroke has occurred. It is relevant to note that these thresholds were observed to be exceeded multiple times during a single stroke, and therefore the detection criteria had to be carefully designed to avoid repeated detection of strokes. This issue was addressed by fixing a time window (composed of $2n$ samples, centered in the stroke detection) in which the stroke can be considered to have been completely executed. Once the stroke has been detected and trimmed, its stored as a set of $6 \times 2n$ samples. The scheme of the detection algorithm is shown in Fig. 3. Notice that k represents the last sample of a detected stroke.

A conservative approach based on a 2 s ($n=20$ samples) interval was considered, with the following values for the thresholds:

$$u_a = 3 \text{ G}$$

$$u_v = 5 \text{ rad/s}$$

According to padel experts, and based on the visualization of the videos recorded during the data collection, it has been considered that 2 s is enough time to prepare and execute the stroke in padel tennis. This detection algorithm was tested on a total of 362 strokes, of which 354 were correctly detected. The accuracy results 97.79%, with a 100% precision, meaning that no false positives are introduced into the data set. For the sake of illustration, in Fig. 4 a comparison between two different types of strokes (forehand and backhand shot) is shown. It is relevant to note the qualitative differences between both types of stroke in the different registered DOF.

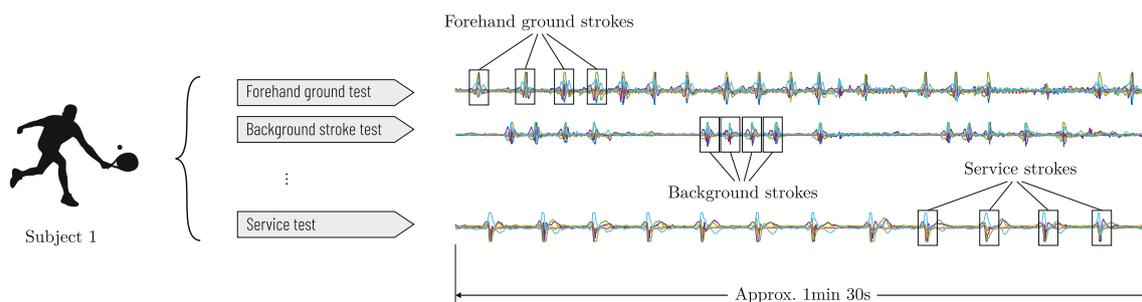


Fig. 2 Scheme of the data capture routine performed for each subject. Note that this structure generates temporal series with strokes of a single type, thus improving the data labeling process

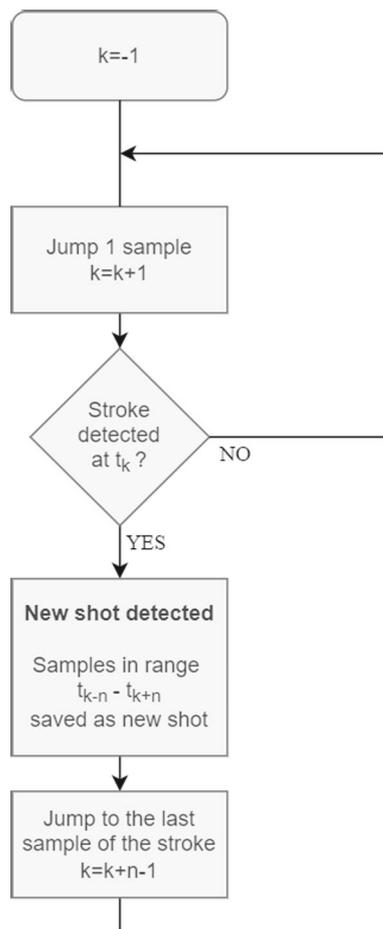


Fig. 3 Stroke detection and trimming from raw data samples

3.4 Target labeling

Once the data related to the shots have been segmented, they should be labeled appropriately to train the machine and deep learning algorithms. The types of shots considered are listed below. For each shot, a brief description has been included to indicate the main goal of the shot. Additionally, in parentheses, the nickname used in the performance evaluation and the

label used in the classification task can be found (see Sect. 5). The considered strokes are as follows:

- **Forehand ground stroke (F, label 0):** The ball is hit on the same side as the skill arm of the player, after the ball bounces off the ground, usually slightly behind the service line.
- **Backhand ground stroke (B, label 1):** It is the opposite stroke to the forehand, as the ball is hit on the opposite side of the skillful arm of the player. It is also performed after bouncing the ball, usually slightly behind the service line.
- **Forehand after the ball bounces off the wall (FW, label 2):** It is a stroke made forward after the ball has contacted the back wall of the court, hitting the ball on the same side of the skilled arm of the player.
- **Backhand after the ball bounces off the wall (BW, label 3):** It is a stroke made forward after the ball has contacted the back wall of the court, hitting the ball on the side opposite the skilled arm of the player. As previously mentioned, this stroke can be performed using a single-hand or a two-hand grip.
- **Forehand Lob (FL, label 4):** It is a defensive stroke that seeks to throw the ball high enough so that the opposing player near the net cannot return the ball to his position and is forced to retreat. The ball is hit on the same side of the player's skilled arm.
- **Backhand Lob (BL, label 5):** It is a defensive stroke that seeks to throw the ball high enough so that the opposing player near the net cannot return the ball from his position and is forced to retreat. The ball is hit on the side opposite to the player's skilled arm.
- **Forehand Lob after the ball bounces off the wall (FLW, label 6):** It is a defensive stroke that seeks to raise the ball so that the opposing player backs up, hitting the ball on the same side of the player's skilled arm. Unlike the forehand lob, in this case the ball is hit after it has contacted the back wall of the court.
- **Backhand Lob after the ball bounces off the wall (BLW, label 7):** It is a defensive stroke that seeks to ele-

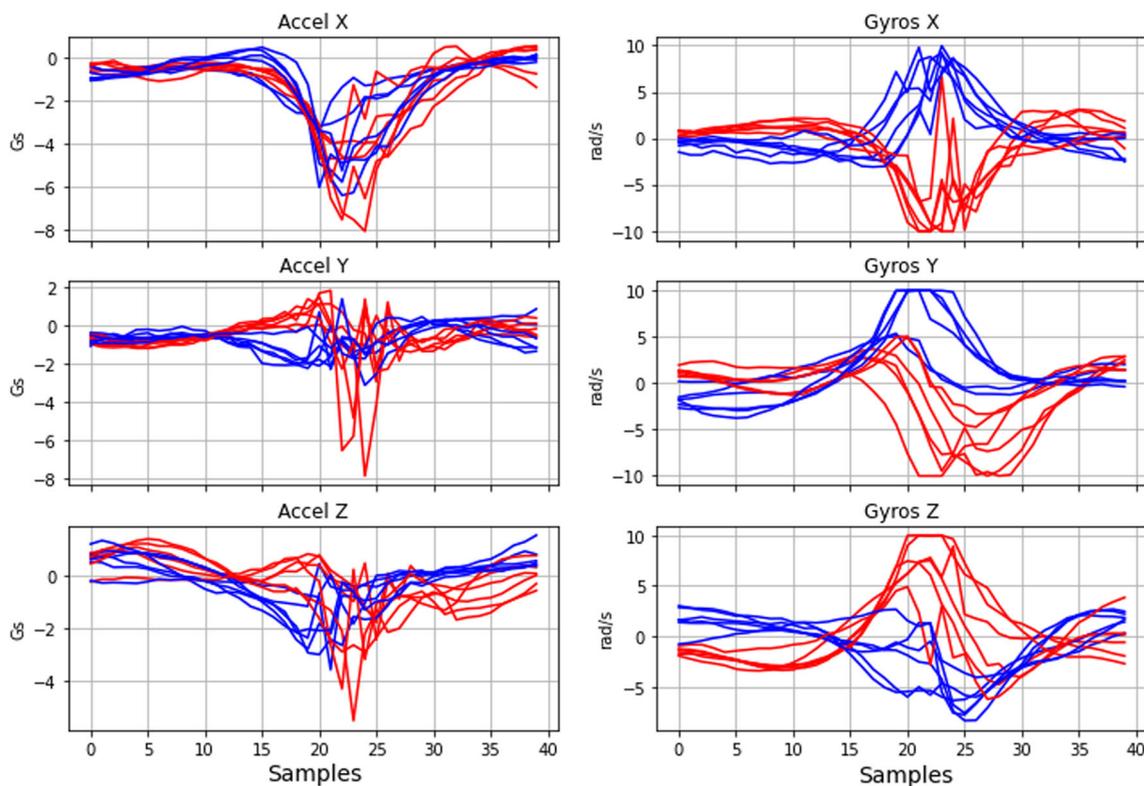


Fig. 4 Comparison between forehand shots (blue) and backhand shots (red)

vate the ball so that the opposing player backs up, hitting the ball on the side opposite to the player’s skilled arm. Unlike the back-hand lob, in this case the ball is hit after it has contacted the back wall of the court.

- **Forehand Volley (FV, label 8):** It is an attacking shot that is played close to the net and without letting the ball bounce. It seeks to keep the opponent at the back of the court and win the point. The ball is hit on the same side of the player’s skilled arm.
- **Backhand Volley (BV, label 9):** It is an attacking shot that is hit close to the net and without letting the ball bounce. It aims to keep the opponent at the back of the court and win the point. The ball is hit from the side opposite to the player’s skilled arm.
- **Tray (T, label 10):** It is a shot halfway between a volley and a smash that is performed in the middle of the court and seeks not to lose the initiative at the net. It is performed on the side of the player’s skilled arm.
- **Smash (S, label 11):** A stroke that is performed by extending the arm upward and hitting the ball at the highest point. It is performed on the side of the player’s skilled arm with the intention of finishing the point.
- **Service (SE, label 12):** It is the stroke that starts the game and is the only one in which the player places the ball himself. Contact with the ball must be at waist level or below and always after bouncing the ball. Unlike ten-

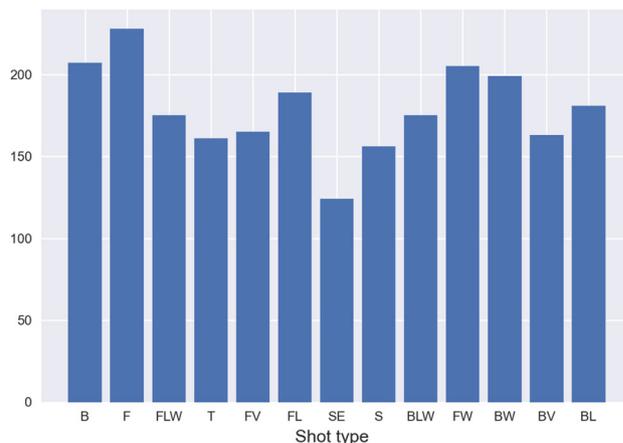


Fig. 5 Number of shots for each category

nis, the aim is not to win the direct point, but to put the opponent in difficulty to gain the attacking position at the net. It is performed on the side of the player’s skilled arm.

Although there are other padel tennis strokes such as off-the-wall smash, *chiquita* or smash for three, it is considered that the previous strokes are the most representative and common in a padel tennis match. The proposed work could easily be extended to consider other padel tennis strokes.

A video recorded during the data collection can be seen in https://www.youtube.com/watch?v=UO_WZ40g1Io. Figure 5 shows the number of shots for each category. It can be observed that the samples are well balanced between the different categories, which is important for a classification problem like the one presented in this work. The minor differences are related to the duration of each shot. For example, in the case of SE, the player should take the ball, prepare, and execute the shot. On the contrary, the F shot is faster since the player hits the ball as it is received from the opponent.

4 Machine and deep learning algorithms

For a first attempt on the classification of padel tennis strokes, in this work six different approaches have been considered, taking into account their different capabilities and performances. Four of these approaches are based on models (both dense and convolutional neural networks, decision tree, support vector machine), while two other are based on instances (K-nearest neighbors and EigenClass). These algorithms were trained using two different types of input: (i) the complete temporal series of strokes and (ii) feature engineering, in which the maximum, minimum, and mean values of each DOF of the strokes were considered. In the former case, the data input consists of 40 timestamps for 6 features (acceleration and speed for each axis, x , y , and z). In the case of feature engineering, the input consists of an array of 18 components (mean, maximum, and minimum for each DOF).

A brief description of each algorithm and the main hyperparameters³ that affect their performance are described in the following subsections.

4.1 Fully connected neural network

Neural networks (NN) represent probably the most popular deep learning algorithms (Voulodimos et al. 2018; Young et al. 2018). NNs are computational models formed by connecting artificial neurons to each other. Each of these is a logic unit that computes a weighted sum of its inputs and then applies a certain activation function, whose result is processed as an output (see Fig. 6-a). Artificial neurons are organized in layers, so that the artificial neurons of a layer receive as input the outputs computed by the previous one, as shown in Fig. 6-b. The softmax layer is responsible for receiving the score of the different classes and computing their probabilities in such a way that they add up to 1.0.

The first deep learning approach proposed in this work is based on a fully connected neural network (FCNN or FC) that receives as input either a temporal series of strokes or

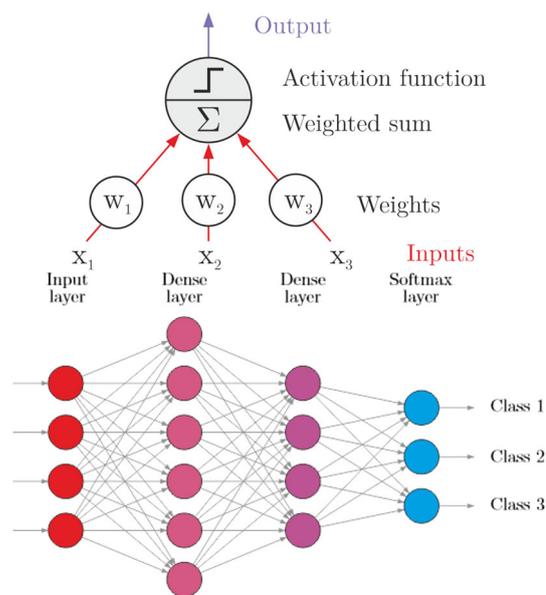


Fig. 6 Example of an artificial neuron with three inputs and a step activation function (top) and a neural network with two dense layers and three output classes (bottom)

the feature engineering data. The last layer has 13 neurons (as many as the classes to be classified).

4.2 1D convolutional neural network

The fully connected neural networks described above have been improved by including 1D-convolutional layers (Ragab et al. 2020; Gutiérrez and Toral 2019). These layers are formed by a reduced set of neurons forming a receptive field that is successively evaluated along the input temporal series. During the learning process, the weights of these neurons are adjusted, forming filters or kernels capable of detecting features along the temporal series, with independence of their position. The resulting 1D convolutional neural network (1DCNN) is based on considering the input data to be a temporal series and therefore is expected to constitute a better classifier for strokes.⁴ Fig. 7 illustrates the performance of a filter through the temporal series in 1DCNN. Each filter passes through the data (simultaneously for each feature) from left to right to generate the output for the next layer.

4.3 Decision tree

A decision tree is a classical machine learning algorithm that builds a tree during training to make predictions (Quinlan 1996; Kumari et al. 2022). This tree contains nested if-else binary conditionals that allow one to approximate the target value. Once the tree is built, it is possible to know the class of

³ The hyperparameters described are the ones used in the hyperparameterization analysis conducted in Sect. 5.

⁴ This algorithm is not suitable for feature engineering input.

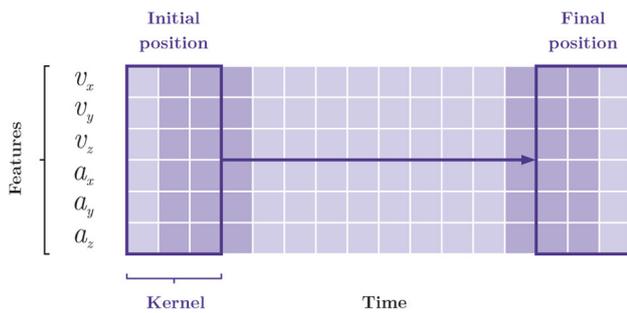


Fig. 7 Illustration of the performance of a filter in a 1DCNN

a new object by passing it through the tree and answering the questions successively. A decision tree is composed of a root node: i) the main node that represents the entire population or data set, ii) the decision or internal nodes, which are used to split the data (make decisions), and iii) the terminal or leaf nodes, nodes where the flow ends and represents the prediction.

4.4 K-nearest neighbors

This algorithm classifies all new data in the corresponding target using the distance between it and the other samples in the training set. The algorithm uses a hyperparameter k to configure the number of neighbors used to predict new data (Cunningham and Delany 2021). Thus, the distance to all neighbors in the training stage is calculated and the most common class among the k nearest neighbors of the new data will be the class assigned to it. Therefore, K-nearest neighbor, KNN, is a simple instance-based algorithm whose performance is given by the value of the parameter k .

4.5 Support vector machine

Support vector machines (SVM) are a set of machine learning algorithms that attempt to find a hyperplane of separation between classes (Burges 1998). The fundamental characteristic of SVMs is the search for an optimal classification that is performed by maximizing the margin of separation between classes, i.e., the hyperplane of separation between classes is equidistant from the closest example of each class.

4.6 Eigenvalue classification

The EigenClass classifier (EC) Erkan (2021) is based on the eigenvalue-based proximity evaluation of the data. This is a novel similarity-based algorithm that has achieved promising results in a number of benchmark classification problems. Further extensions of this algorithm can be found in (Memiş et al. 2022; MEMİŞ et al. 2022; Memiş et al. 2021). The algorithm uses a hyperparameter k , which refers to the

number of eigenvalues employed in the proximity determination.

5 Performance evaluation

This section presents the evaluation of the machine and deep learning algorithms used for the padel stroke target classification problem. First, a hyperparameter tuning must be performed to compare the algorithms with the best possible configuration for the target classification problem. Second, the algorithms are compared under the best configuration. Algorithms have been developed in Python using the Keras⁵ and Scikit-learn⁶ libraries. The data set and the codes used in the work are publicly available at <https://github.com/guillecarter/Padel-Shot-Classification-and-Dataset>, with the exception of the EigenClass algorithm, which is available at the algorithm author's repository <https://www.mathworks.com/matlabcentral/fileexchange/78462-eigenvalue-classification-for-machine-learning>.

5.1 hyperparameters setting

Hyperparameter tuning is essential for unbiased comparison of algorithms. Therefore, a grid analysis for each algorithm has been developed considering the most important parameters. The data set has been divided into three parts for training (64%), validation (16%), and test (20%). The performance metric used to evaluate the approaches is accuracy, as it reflects the percentage of shots classified appropriately. The accuracy can be calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

where:

- TP (True Positive): The algorithm predicts that an observation belongs to a class and that it does belong to that class.
- TN (True Negative): The algorithm predicts that an observation does not belong to a class and that it does not belong to that class.
- FP (False Positive): The algorithm predicts that an observation belongs to a class and it does not belong to that class.
- FN (False Negative): The algorithm predicts that an observation does not belong to a class and that it does belong to that class.

⁵ <https://keras.io/>.

⁶ <https://scikit-learn.org/stable/>.

The accuracy metric is suitable for the proposed classification since we assign the same importance for classifying each shot. Therefore, we are interested in knowing the general capacity of machine and deep learning algorithms for the classification task. In the following subsections, the accuracy of each algorithm will be evaluated for different hyperparameter values. For each configuration, 15 runs have been conducted and boxplots are employed to show the distribution of the obtained results. In addition, in some cases, the confusion matrix will be shown for further explanation of the results. The data used correspond to the training and validation sets.

5.1.1 Fully connected neural networks

The number of layers and neurons for each layer have been the hyperparameters considered for this case. In addition, training parameters such as the number of epochs and batch size were also considered in the analysis carried out. Hyperparameterization was performed using a grid search, with the values summarized in Table 2. Notice that only the number of neurons in the first layer is indicated, and the following layers (in case of more than one layer) use half the number of neurons in each case.

The results obtained for each configuration are shown in Fig. 8. According to the results, the best configuration corresponds to an architecture with three dense layers (TS3L in Fig. 8). It is observed that when the input is the temporal series (TS in Fig. 8) the classifier response is considerably better than when the input is with the feature engineering technique (FE in Fig. 8). In fact, the feature engineering classifier has about 80% accuracy, while the temporal series classifier is close to 90%. Therefore, the temporal series input represents a 10% improvement in accuracy over the feature-engineered classifier. For the dense neural network, the best hyperparameterization is one with three layers, a first layer with 1000 neurons, followed by a second with 500 and a third with 13 (one per class), with 70 epochs and 30 batch size, which achieves a maximum accuracy of 92.06%. The average is 90.44% ($\pm 0.57\%$).

Figure 15a shows the confusion matrix obtained for the FC architecture with three dense layers with input from the temporal series. It can be seen that the performance of the algorithm is suitable for each stroke. However, some errors can be found in some forehand strokes (with/without using the wall) and also between the tray and the smash (T and S in Fig. 15a). Note that some novice players may have difficulties executing the tray; therefore, it can be expected in the data set that some players executed those strokes inappropriately.

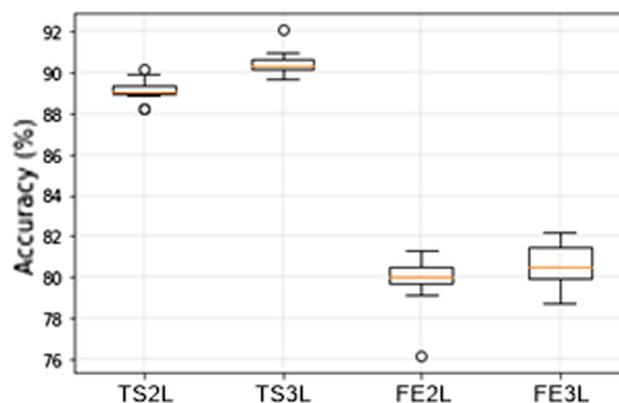


Fig. 8 FC algorithm hyperparameter results (Temporal Series TS, Feature Engineering FE, Layer L)

5.1.2 1D convolutional neural networks

In this algorithm, the hyperparameters that are evaluated are the number of filters and the filter size. As in the fully connected case, the number of epochs and batch size were also considered. For obvious reasons, this classifier has not been tested with the feature engineering technique. Hyperparameterization was carried out by means of a grid search, with the parameters summarized in Table 3. The proposed configurations included one and two convolutional layers attached to one and two dense layers. For 1DCNN with two convolutional layers, the number of filters on the second was set to half the number for filters of the first one. Also, after the convolutional layer/s, a dropout and a maxpooling layer were included.

Figure 9 shows the distribution of the results obtained for the 1DCNN classifiers. The configuration that achieves the highest accuracy is the one with one convolutional layer and two dense layers (1C+2D in Fig. 9), achieving 93.35% for its highest accuracy, with a mean of 91.28%. However, the dispersion of this model is high ($\pm 2.41\%$). The first layer is a convolutional layer of 256 filters, with a kernel of 8. It is followed by a dropout layer and a maxpooling layer. Then it has a flatten layer prior to the densely connected layer with 1000 filters and with ReLU activation function, and another dense layer of 13 filters and with softmax activation function.

On the other hand, the 2C+1D configuration gathers a high accuracy with a lower dispersion, with 91.08% ($\pm 1.01\%$). In this case, the first convolutional layer has 256 filters, while the second has 128, both with a kernel of 8. It should be noted that the four options represented are very close to each other, around 90% in all cases, which implies a considerably correct response of these models.

Figure 15b shows the confusion matrix for the best configuration of the 1D convolutional network. The results are satisfactory for all the considered strokes. Only some mistakes can be found in forehand strokes. For example, FW

Table 2 Hyperparameterization of fully connected neural networks tested

Input type	Temporal series	Feature engineering
Layers	[2,3]	[2,3]
Neurons on first layer	[10,100,200,1000]	[100,500,1000,1500,2000]
Epochs	[10,40,70]	[40,70,100,200]
Batch size	[30,50,70]	[30,50,70]

Table 3 Hyperparameterization of convolutional neural networks tested

Configuration	1C+1D	1C+2D	2C+1D	2C+2D
Filters on 1 st conv. layer	[32,64,128,256]	[32,64,128,256]	[32,64,128,256]	[32,64,128,256]
Kernel size	[3,5,8]	[3,5,8]	[3,5,8]	[3,5,8]
Filters on 1 st dense layer	–	[100,1000]	–	[100,1000]
Epochs	[40,70]	[40,70]	[40,70]	[40,70]
Batch size	[10,30,50,70]	[10,30,50,70]	[10,30,50,70]	[10,30,50,70]

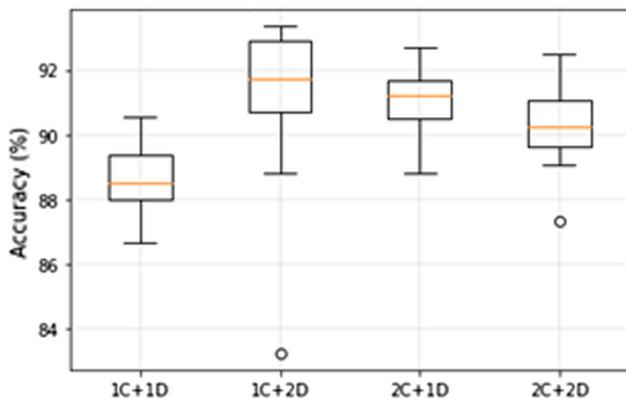


Fig. 9 Convolutional 1D algorithm hyperparameter results (Convolutional layer C, Dense layer D)

and FWL are confused. Notice that these are similar strokes that are executed after the ball bounces off the wall. Thus, depending on the high of the ball, the player can have some difficulties for executing the lob in a suitable position. Also, as in FCNN there are some problems in distinguishing between T and S.

5.1.3 Decision tree

The parameters used in the grid search for the decision tree are summarized in Table 4. The same parameters were considered for both temporal series and feature engineering input data.

Regarding the hyperparameter for the temporal series input, the best hyperparameter configuration is the one with a tree depth of 40 (*max_depth*), at least 4 samples to split a node (*min_samples_split*), at least one sample per leaf (*min_samples_leaf*) and entropy impurity function. The classifier obtains an accuracy of 60.59% ($\pm 1.06\%$). In terms

Table 4 Hyperparameterization of the decision tree

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]

of feature engineering, the best hyperparameter configuration is the one with a tree depth of 10 (*max_depth*), at least 2 samples to split a node (*min_samples_split*), at least one sample per leaf (*min_samples_leaf*) and entropy impurity function. The classifier obtains an accuracy of 60.76% ($\pm 0.90\%$).

Figure 10 shows a summary of the results obtained for the decision tree classifiers. The temporal series classifier achieves a mean accuracy of 60.59%, compared to 60.76% for the feature engineering classifier. Both mean values are very close, however, the classifier with feature engineering has a lower dispersion. Regarding the maximum value, the temporal series input achieves a higher accuracy, reaching 62.09%.

Figure 15c shows the confusion matrix for the best decision tree configuration using feature engineering. It can be seen that the algorithm fails to classify some strokes. It is noticeable that the algorithms struggle to discern between the strokes using or not using the wall for both cases forehand (F and FW in Fig. 15c) and (B and BW in Fig. 15c) backhand. Also, there are problems in classifying the backhand stroke and the backhand volley. Regarding the tray T and the smash S, the errors have increased significantly with respect to the neuronal networks (from 3 to 13).

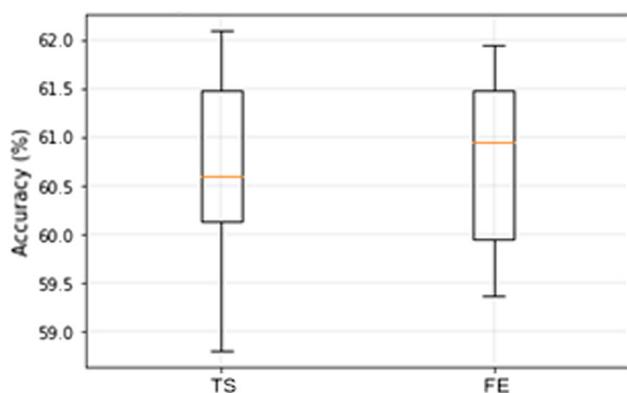


Fig. 10 Decision tree algorithm hyperparameter results (Temporal Series TS, Feature Engineering FE)

Table 5 Hyperparameterization of K values in KNN for temporal series input

k	1	2	3	4	5
	84.12	79.97	80.97	79.26	78.83

Table 6 Hyperparameterization of K values in KNN for feature engineering input

k	1	2	3	4	5
	68.24	64.95	65.38	64.66	64.38

5.1.4 K-nearest neighbors

In this algorithm, the number of neighbors k used to make the predictions is the hyperparameter considered. Table 5 contains the accuracy obtained for different values of k for temporal series input. It can be seen that, in general, as the value k increases, the performance of the algorithm worsens (Tables 5 and 6).

Table 6 includes the results when feature engineering is used. Again, the best performance is obtained with $k = 1$.

Figure 11 shows the best configuration of the KNN algorithm ($k = 1$) for both feature engineering and temporal series inputs. It can be observed that the KNN algorithm presents significantly better results when the input is a temporal series. In fact, the accuracy achieved is higher than 80%, which is a good performance for a classification problem of 13 categories like the one targeted in this work.

Figure 15d shows the confusion matrix for the KNN algorithm using temporal series. It can be observed that, as in the previous algorithm, there are some problems between the same strokes using or not the wall (for instance, between the backhand lob and the backhand lob after the ball bounces off the wall).

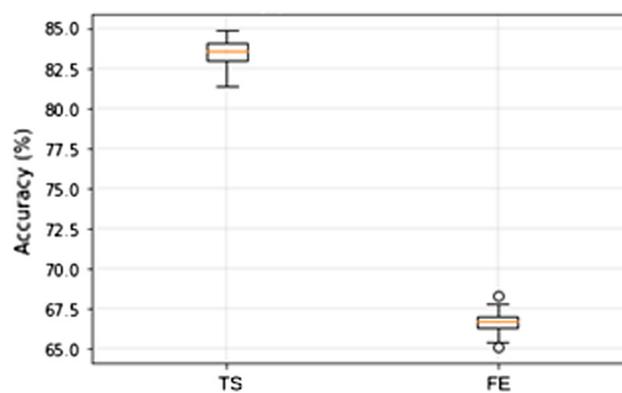


Fig. 11 KNN algorithm hyperparameter results (Temporal Series TS, Feature Engineering FE)

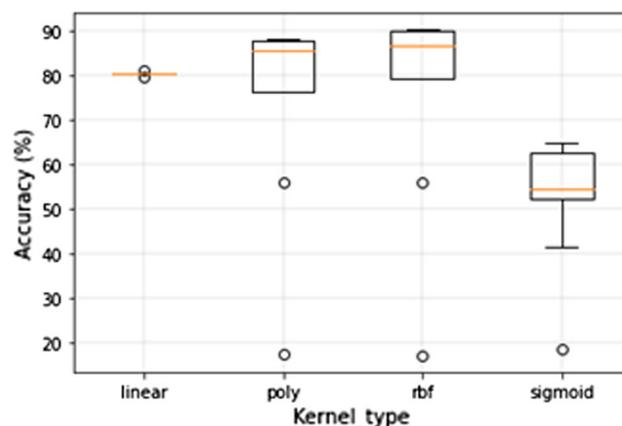


Fig. 12 SVM algorithm hyperparameter results with temporal series input

5.1.5 Support vector machines

The hyperparameterization of the algorithm was based on the type of kernel and the regularization C , attending to the accuracy. The values considered are summarized in Table 7

Table 8 contains the accuracy achieved by the SVM according to the hyperparameters used for the input of the temporal series. According to the results, the best combinations are $C = 10$ or $C = 100$ with a kernel of type RBF.

Table 9 includes the hyperparameter analysis for the feature engineering method. In this case, the best configuration is $c = 100$ with a kernel of type RBF. Notice that the results are significantly worse when feature engineering is used.

Figure 12 shows the distribution of the results for the best configuration of c for each kernel when the algorithm input is a temporal series. In the case of the RBF kernel, the results are satisfactory, achieving an accuracy of more than 90%; therefore, the SVM algorithm is only outperformed by IDCNN. Similarly, Fig. 13 shows the results for the case of feature engineering. In general, the results get worse when feature engineering input is used in the SVM algorithm.

Table 7 Hyperparameterization of the support vector machine

Input	Temporal series	Feature engineering
Kernel filter	[Linear, Polynomial, Radial, Sigmoid]	[Linear, Polynomial, Radial, Sigmoid]
Regularization (C)	[0.01, 0.1, 0.5, 1, 2, 10, 12, 20, 100]	[0.01, 0.1, 1, 10, 100, 200, 300, 500, 1000]

Table 8 SVM accuracy (%) with temporal series, in terms of kernel and regularization C

		Regularization (C)								
		0.01	0.1	0.5	1	2	10	12	20	100
Kernel	Linear	81.40	79.83	80.40	80.40	80.40	80.40	80.40	80.40	80.40
	Polynomial	17.45	56.22	76.39	82.69	85.69	88.27	88.13	88.13	88.13
	Radial	17.17	55.94	79.40	85.84	86.98	90.41	90.27	89.99	90.41
	Sigmoid	18.60	41.63	62.80	65.09	62.66	55.22	53.79	54.51	52.22

Table 9 SVM accuracy (%) with feature engineering, in terms of kernel and regularization C

		Regularization (C)								
		0.01	0.1	1	10	100	200	300	500	1000
Kernel	Linear	62.66	68.81	70.82	69.96	69.67	70.10	70.10	69.38	69.53
	Polynomial	23.46	44.64	61.52	68.96	73.82	72.82	73.24	72.53	71.39
	Radial	9.73	40.77	61.37	68.53	74.25	73.96	72.82	72.96	71.24
	Sigmoid	9.73	18.45	31.62	29.47	23.46	23.75	22.89	22.75	24.18

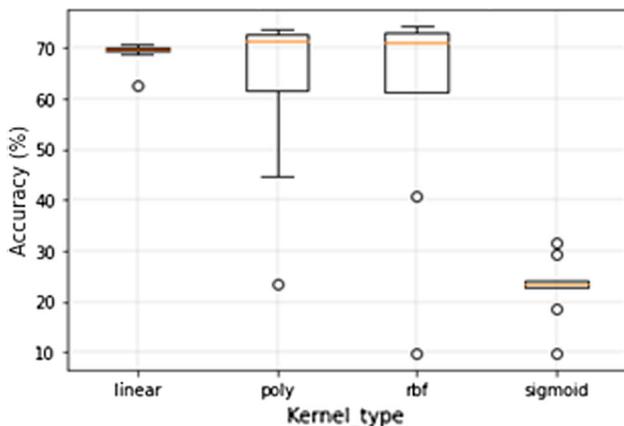


Fig. 13 SVM algorithm hyperparameter results with feature engineering input

Figure 15e shows the confusion matrix for the best configuration of SVM with input from the temporal series. In this case, the most significant errors are found between BLW and BL. Therefore, again some problems are presented to discern between shots that use the wall with those that do not use it.

5.1.6 Eigenvalue classification

For this algorithm, the hyperparameter k has been considered for the algorithm tuning. Table 10 contains the accuracy obtained for different values of k for temporal series input.

Table 10 Hyperparameterization of k values in EC for feature engineering input

k	1	2	3	4	5
	65.04	67.91	67.77	66.48	65.76

Table 11 Hyperparameterization of k values in EC for temporal series input

k	1	2	3	4	5
	64.47	65.62	64.04	62.61	60.60

It can be seen that, in general, as the value k increases, the performance of the algorithm worsens.

Table 11 includes the results when feature engineering is used. Again, the best performance is obtained with $k = 1$.

Figure 14 shows the best configuration of the EC algorithm ($k = 2$) for both feature engineering and temporal series inputs. It can be observed that this algorithm presents significantly better when the input is feature engineering, with an achieved accuracy close to 70%, which is a good performance for a classification problem of 13 categories like the one targeted in this work, and noticeably higher than the decision tree. The confusion matrix for the EC algorithm using feature engineering is shown in Fig. 15f.

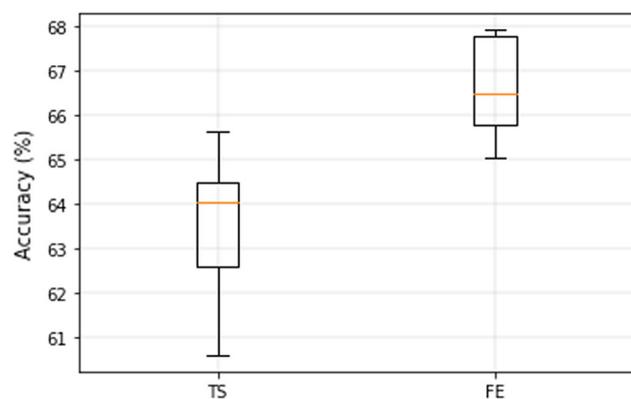


Fig. 14 EC algorithm hyperparameter results (Temporal Series TS, Feature Engineering FE)

5.2 Performance comparison

The results obtained are summarized in Table 12, where it can be seen that the 1D CNN can achieve up to 93.35% accuracy. Precision, recall, and f1-score macro-average are also shown for the sake of completeness in the comparison. It is also relevant to highlight that only the results for temporal series input are shown, since they achieved far better performances than the feature engineering cases. The accuracy of the best hyperparameterization of each algorithm is shown in Fig. 16

The most immediate result is the poor performance of the feature engineering technique, which does not provide any improvement for any of the algorithms studied, with an average accuracy loss of 16% compared to the temporal series classifier. This could be explained by the fact that the characteristics extracted from the temporal series to apply feature engineering (maximum, minimum, and average values) are not the most representative for the classification of the strokes, or simply because the algorithm has a better response to the temporal series, as it inputs more information.

However, it is interesting that the 1DCNN is not clearly superior to the rest, despite being a classifier specifically designed to receive temporal series as input. In fact, the neural network without convolution is only 1% below, and thus CNN did not provide such a significant improvement. This may be because all the hits are tightly framed within the time window in which they are collected, so that all the algorithms already receive the hits fairly well aligned, that is, each hit always has its maximum or minimum values at the center of the hit, and the detection algorithm is actually responsible for the task at which the CNNs perform best.

Table 12 also shows the average training and prediction time for each algorithm evaluated. The computer used for the simulations is an Intel(R) Core(TM) i7-4510U CPU@2.00GHz, 16GB RAM, and NVIDIA GeForce 840 M. According to the results, the 1DCNN algorithm requires more computational time for both training and prediction.

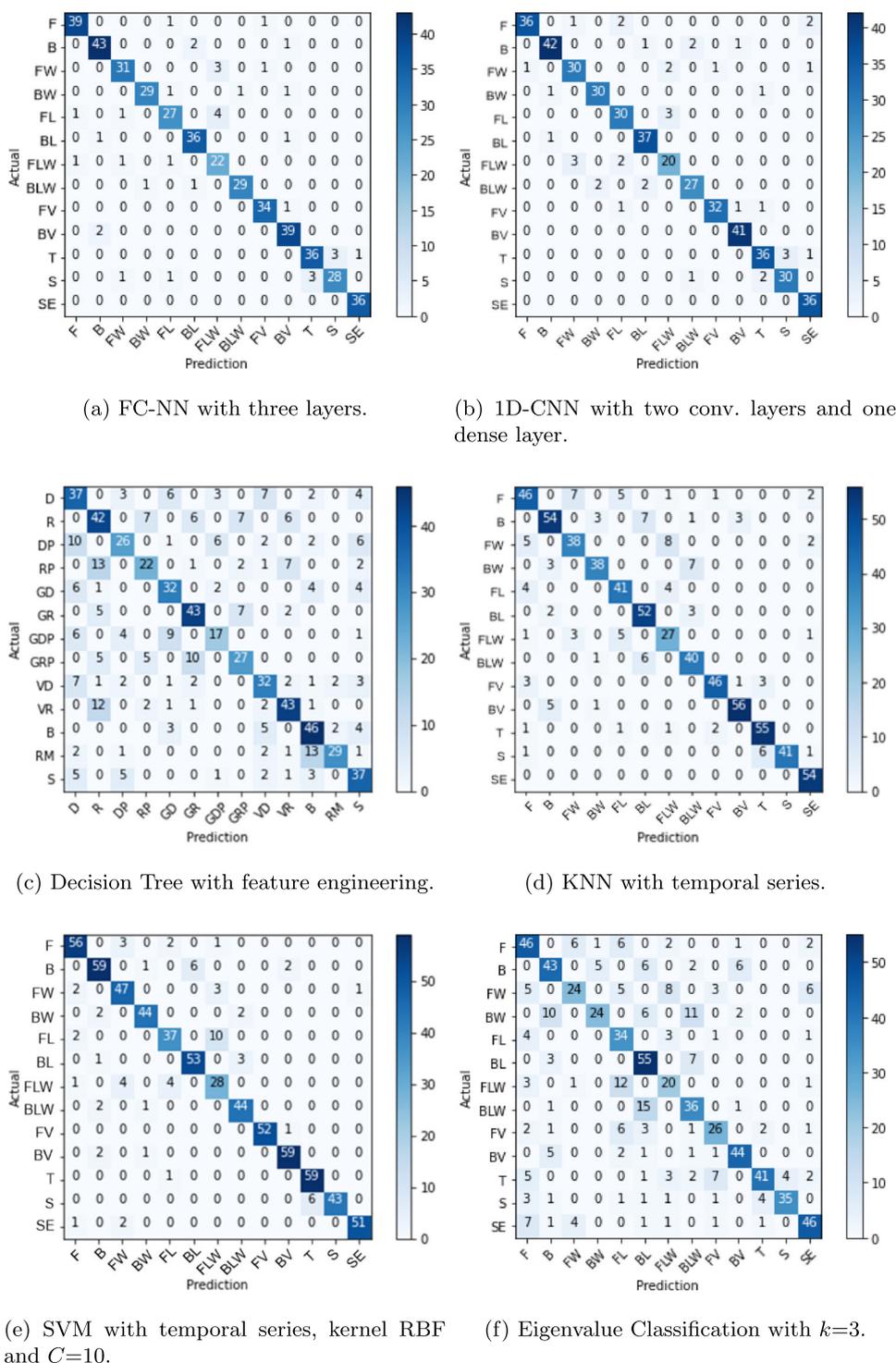
However, the prediction time is very low (much less than one second) for all the algorithms evaluated. Therefore, the evaluated algorithms can be implemented in real-time scenarios.

5.3 Discussion of the results

The aim of this study was to compare some of the main machine learning and deep learning algorithms in the detection and classification of padel tennis strokes. In general, the temporal series input is more suitable than the feature engineering input for the machine learning and deep learning algorithms evaluated. Only the decision tree works better with feature engineering. Therefore, it is demonstrated the classification of padel strokes cannot be properly done with simple statistics of the temporal series, such as the mean, variance, and maximum and minimum values. Therefore, more sophisticated metrics should be used like the ones used for machine and deep learning algorithms. It is also important to note that the feature engineering used is simple, so more sophisticated approaches can be considered in the future. As shown by Table 12, the best algorithm is 1DCNN with temporal series, which achieves an accuracy of 93.35% with a neural network architecture of 1 convolutional layer followed by two dense layers. Fully connected networks also exhibit very good performance on the target task with input from the temporal series. We believe that such high performance is due to the fact that the segmentation procedure executed during the data set creation makes possible a suitable data alignment for a fully connected architecture. In case such an alignment is not as good as the one presented in this work, higher differences could be expected between the convolutional and fully connected networks. KNN with temporal series input is a simple algorithm that presents moderate results. SVM algorithm with input of temporal series and an RBF configuration and $C = 10$ or $C = 100$ achieves significant results that are only outperformed by 1DCNN. In addition, the SVM presents lower time complexity for training and prediction. Therefore, KNN and SVM can be good candidates in scenarios with low computational resources.

As shown in Fig. 15a, 15b, 15c, 15d, and 15e, some algorithms have problems to classify certain strokes. For instance, forehand ground stroke and forehand after the ball bounces off the wall seem to be likely to be misclassified, and some errors are also present to discern between the tray stroke and the smash. Previous works in racket sports have also found difficult to separate similar shots due to the similar shape of the signals from the sensors' data (Tabrizi et al. 2020; Ebner et al. 2019; Whiteside et al. 2017). There are two factors that might explain this. First, the variety of stroke-execution possibilities might have created some crossover in the identification of signals. Most research carried out in this line has developed algorithms to classify between three and four types

Fig. 15 Summary of confusion matrices for the methods evaluated



of stroke (Benages Pardo et al. 2019; Kos and Kramberger 2017; Tabrizi et al. 2020), and to the best of our knowledge, few studies have attempted to discriminate shots with a similar level of detail (Whiteside et al. 2017). It must be pointed out that in our study we aimed to classify up to 13 different strokes. This fact required the algorithm to distinguish and classify strokes that differ only in a few nuances, which is

actually the second potential explanation for the misclassification problems encountered in the study. Despite this, it should be noted that the majority of the algorithms which have been evaluated achieved a good performance.

Overall, this study provides promising findings for an interesting research avenue: the use of bio-sensing technology to improve skills training in sports. A recent study has

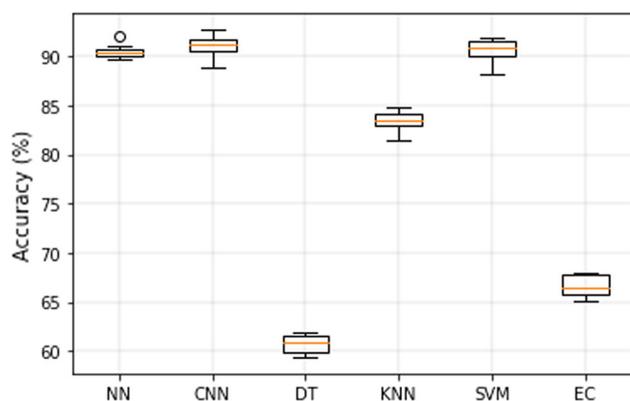


Fig. 16 Accuracy for the best hyperparameterization of each algorithm

compared traditional teaching and teaching through wearable devices in badminton (Lin et al. 2021). The results obtained in this quasiexperimental study have shown how such technology can benefit practitioners training through to the provision of instant and adaptive feedback in motor skill learning.

6 Conclusion and further works

The work presented in this paper involves the creation of the first padel tennis stroke data set and contributes to the existing literature with the first classification of padel tennis strokes. The findings of algorithms comparison, summarized in Table 12, shed some light on the functioning of different algorithms in the detection and identification of racket sports strokes. For this purpose, a first data set has been created by gathering data from 12 different players, with skill levels ranging from amateur to professional, with more than 2300 different strokes of 13 different types. The raw data con-

sisted of temporal series of 6 DOFs (3 linear accelerations and 3 angular rotations) measured at the player's wrist during the execution of the strokes. The data set was built from the raw data by means of an algorithm designed to identify and trim out the strokes from the time data series, on the basis of acceleration and angular velocity thresholds. The data set was used to train a total of 6 different machine and deep learning algorithms (FCNN, 1DCNN, DT, KNN, EC and SVM), whose performances were later compared. Among the 6 approaches, the 1DCNN exhibited the best performance, reaching an accuracy of 93% with one convolutional layer and two dense layers. In sight of the goodness of the proposed approaches, this work is proposed to continue along several lines of improvement. Firstly, in the present study, and in line with most research in the field (Blank et al. 2015; Kos and Kramberger 2017; Kos et al. 2016; Srivastava et al. 2015; Whiteside et al. 2017), the sensor was strapped on the wrist of the dominant arm. In light of relatively recent evidence (Ebner et al. 2019), it might be worth exploring whether using the racket as the sensor placement could improve the classification system. The second line of improvement consists of increasing the size of the database, including left-handed players. Moreover, and given the cost and extensive labor involved in the data capture routines, data augmentation is proposed to generate new data. The third improvement proposed is thus to extend the database features to differentiate between the different effects that the different strokes provide to the ball, for which it will be necessary to collect new data in which only one type of effect is performed in each test. Another research direction is to consider the frequency of the acceleration and speed signals to perform the target classification in the frequency domain. Finally, the performance comparison can be extended with other machine learning algorithms (Memiş et al. 2022; MEMİŞ et al. 2022; Memiş et al. 2021).

Table 12 Summary of parameters used for machine learning algorithms, achieved test results and average training and prediction times

Classifier	FCNN	1D-CNN	DT	SVM	K-NN	EigenClass
Hyperparameters	3 layers (1000, 500, 13 n.) 70 epochs 30 batch size	1 conv. layer (256 filt.) 2 dense layers (1000, 13 filt.) 70 epochs 70 batch size	40 max depth Min samples split 4 Min samples leaf 1 Entropy criterion	Kernel RBF C=10	k=1	k=3
Accuracy	92.6	93.35	62.09	91.85	84.84	67.91
Precision	92.19	92.30	62.57	91.72	85.46	69.35
Recall	91.54	91.63	62.22	91.14	84.73	67.26
F1-score	91.79	91.82	62.18	91.26	84.75	67.18
Training time (s)	26.05	98.66	5.87	0.22	0.003	15.51
Prediction time (ms)	42.84	49.11	0.21	0.65	2.11	0.003

Acknowledgements The authors would like to thank Padel Sport Home and all the players that have participated in the data collection.

Funding Funding for open access publishing: Universidad de Sevilla/CBUA The authors did not receive support from any organization for the submitted work.

Availability of Data and Materials The data sets generated during the current study are available from the authors on reasonable request.

Declaration

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval This article does not contain any study with human participants or animals performed by any of the authors.

Authorship Contribution statement All the authors have equal contributions in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

A Hyperparameters of machine learning and deep learning algorithms

This appendix describes the hyperparameters used in each algorithm in the performance comparison.

A.1 Fully connected network

The main parameters that must be configured in an FC are⁷:

- **Number of layers:** This is the number of layers used in the neuronal network. It determines the depth and complexity of the network.
- **The number of neurons:** Indicates the number of neurons used in each layer. Normally, the number of neurons used is reduced as the data go through the input layer onward.

⁷ Notice that training parameters are not considered in this brief description such as activation function, optimizers, batch size, and number of epochs, among others.

A.2 1D convolutional neural networks

Three main hyperparameters determine the performance of 1DCNNs:

- **Number of layers:** Number of layers used in the neural network.
- **Number of filters:** Determine the number of filters used in each layer. The higher the number of filters, the higher the complexity of the neuronal network.
- **Filter size:** Indicates the number of time steps of the input in each layer that are used to pass through the filters.

A.3 Decision tree

The main parameters considered that determine the performance of decision trees are as follows:

- ***max_depth*:** Indicates how deep the tree can be. The deeper the tree, the more splits it has, and it captures the more information about the data. Care should be taken since high depth tends to lead to overfitting.
- ***min_samples_split*:** It represents the minimum number of samples required to split an internal node. This can vary between considering at least one sample at each node and considering all samples at each node. When this parameter is increased, the tree becomes more constrained because it has to consider more samples at each node.
- ***min_samples_leaf*:** The minimum number of samples required to be in a leaf node. This parameter is similar to *min_samples_splits*; however, this describes the minimum number of samples at the leaf base, the base of the tree.
- ***criterion*:** The function to measure the quality of a split. In this work, two criteria will be evaluated, Gini impurity and entropy.

A.4 Support vector machines

The main parameters that determine the performance of the SVM algorithm are:

- ***kernel*:** It is a function that is used to transform (*tricky kernel*) the data to a new hyperplane that helps with the classification of the samples. The most common kernels are: lineal, polynomial, Radial Basic Function (RBF), and sigmoid.
- **Regularization *C*:** It controls the regularization on the hyperparameter *C*. The regularization consists of generalizing the model for most of the cases, even if some training cases are not perfectly classified. The value of

C is inversely proportional to the strength of the regularization.

A.5 K-nearest neighbors

The main parameter that determines the performance of the KNN algorithm is:

- K : It is the number of nearest neighbors that are considered to perform the classification.

A.6 Eigenvalue classification

The main parameter that determines the performance of the EC algorithm is:

- k : It is the number of eigenvalues that are considered in the determination of the distance metric.

References

- Blank P, Hoßbach J, Schuldhau D, Eskofier BM (2015) Sensor-based stroke detection and stroke type classification in table tennis. In Proceedings of the 2015 ACM International Symposium on Wearable Computers, pages 93–100
- Burges Christopher JC (1998) A tutorial on support vector machines for pattern recognition. *Data Min Knowl Dis* 2(2):121–167
- Cunningham P, Delany SJ (2021) K-nearest neighbour classifiers-a tutorial. *ACM Comput Survys (CSUR)* 54(6):1–25
- Ebner CJ, Findling RD (2019) Tennis stroke classification: comparing wrist and racket as imu sensor position. In Proceedings of the 17th international conference on advances in mobile computing & multimedia, pages 74–83
- Erkan U (2021) A precise and stable machine learning algorithm: eigenvalue classification (eigenclass). *Neural Comput Appl* 33(10):5381–5392
- Gutiérrez D, Toral S (2019) Deep neuronal based classifiers for wireless multi-hop network mobility models. In: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pages 602–607. IEEE
- Kos M, Kramberger IZ (2017) A wearable device and system for movement and biometric data acquisition for sports applications. *IEEE Access* 5:6411–6420
- Kos M, Ženko J, Vlaj D, Kramberger I (2016) Tennis stroke detection and classification using miniature wearable imu device. In: 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), pages 1–4. IEEE
- Kumari LV, Padma Sai Y et al (2022) Classification of ecg beats using optimized decision tree and adaptive boosted optimized decision tree. *Signal, Image and Video Proc* 16(3):695–703
- Ladany SP, Machol RE (1977) *Optimal strategies in sports*, volume 5. Elsevier Science Limited
- Lin K-C, Wei C-W, Lai C-L, Cheng I, Chen NS, et al. (2021) Development of a badminton teaching system with wearable technology for improving students' badminton doubles skills. *Educ Technol Res Dev*, 69(2):945–969
- Martínez Bernardino Javier Sánchez-Alcaraz (2013) Historia del pádel= history of padel. *Materiales para la historia del deporte* 11:57–60
- McGrath J, Neville J, Stewart Tom, Cronin J (2021) Upper body activity classification using an inertial measurement unit in court and field-based sports: A systematic review. *Proc Instit Mech Eng, Part P: J Sports Eng Technol* 235(2):83–95
- Memiş S, Enginoğlu Serdar, Erkan U (2022) A classification method in machine learning based on soft decision-making via fuzzy parameterized fuzzy soft matrices. *Soft Comput* 26(3):1165–1180
- MEMİŞ S, ENGİNOĞLU S, ERKAN U (2022) A new classification method using soft decision-making based on an aggregation operator of fuzzy parameterized fuzzy soft matrices. *Turkish J Electr Eng Comput Sci* 30(3):871–890, 2022
- Memiş S, Enginoğlu S, Erkan U (2021) Numerical data classification via distance-based similarity measures of fuzzy parameterized fuzzy soft matrices. *IEEE Access* 9:88583–88601
- Mems sensor market analysis by type (2021) (mechanical, optical, chemical & biological, thermal mems sensors), by fabrication material (silicon, polymer, ceramic, metal mems sensors), by application, by region - global forecast 2022-2032. Technical Report FACT4528MR, FACTMR
- Pardo LB, Perez DB, Urunuela CO (2019) Detection of tennis activities with wearable sensors. *Sensors*, 19(22):5004
- Quinlan JR (1996) Learning decision tree classifiers. *ACM Comput Survys (CSUR)* 28(1):71–72
- Ragab MG, Abdulkadir SJ, Aziz N (2020) Random search one dimensional cnn for human activity recognition. In: 2020 International Conference on Computational Intelligence (ICCI), pages 86–91. IEEE
- Ramón L, José G, Salvador L, Goran V, Diego M, Sánchez Alcaraz Martínez Bernardino. Análisis de la distancia recorrida en pádel en función del nivel de juego y el número de puntos por partido (analysis of distance covered in padel based on level of play and number of points per match). 39
- Ramón-Llín J, Guzmán JF, Muñoz D, Martínez-Gallego R, Sánchez-Pay A, Sánchez-Alcaraz BJ (2021) Análisis secuencial de golpes finales del punto en pádel mediante árbol decisional analysis of shots patterns finishing the point in padel through decision-tree analysis
- Seshadri DR, Li RT, Voos JE, Rowbottom JR, Alfes CM, Zorman CA, Drummond CK (2019) Wearable sensors for monitoring the physiological and biochemical profile of the athlete. *NPJ Dig Med* 2(1):1–16
- Sha X, Wei G, Zhang X, Ren X, Wang S, He Zhonghai, Zhao Y (2021) Accurate recognition of player identity and stroke performance in table tennis using a smart wristband. *IEEE Sensors J* 21(9):10923–10932
- Sporting Intelligence (2015) *Global sports salaries survey 2015*. Päivitetty,
- Srivastava R, Patwari A, Kumar S, Mishra G, Kaligounder L, Sinha P (2015) Efficient characterization of tennis shots and game analysis using wearable sensors data. In 2015 IEEE sensors, pages 1–4. IEEE
- Tabrizi SS, Pashazadeh S, Javani V (2020) Comparative study of table tennis forehand strokes classification using deep learning and svm. *IEEE Sensors J* 20(22):13552–13561
- Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) *Deep learning for computer vision: A brief review*. *Comput Intell Neurosci* 2018
- Whiteside D, Cant O, Connolly Molly, Reid M (2017) Monitoring hitting load in tennis using inertial sensors and machine learning. *Int J Sports Physiol Perform* 12(9):1212–1217
- Wu M, Fan M, Hu Y, Wang R, Wang Y, Li Y, Wu S, Xia G (2022) A real-time tennis level evaluation and strokes classification system based on the internet of things. *Internet of Things*, page 100494
- Yang B, Cheng B, Liu Y, Wang L (2021) Deep learning-enabled block scrambling algorithm for securing telemedicine data of table tennis players. *Neural Comput Appl*, pages 1–14

- Young T, Hazarika D, Poria S, Cambria E (2018) Recent trends in deep learning based natural language processing. *IEEE Comput Intell Magaz* 13(3):55–75
- Zhou H, Gan Yu (2021) Research on pedestrian detection technology based on the svm classifier trained by hog and ltp features. *Future Gener Comput Syst* 125:604–615

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.