Data Min Knowl Disc
DOI 10.1007/s10618-017-0495-0

Activity recognition in beach volleyball using a Deep Convolutional Neural Network

Leveraging the potential of Deep Learning in sports

Thomas Kautz[1] · Benjamin H. Groh[1] · Julius Hannink[1] · Ulf Jensen[1] · Holger Strubberg[2] · Bjoern M. Eskofier[3]

Received: 29 February 2016 / Accepted: 8 February 2017

Abstract Many injuries in sports are caused by overuse. These injuries are a major cause for reduced performance of professional and non-professional beach volleyball players. Monitoring of player actions could help identifying and understanding risk factors and prevent such injuries. Currently, time-consuming video examination is the only option for detailed player monitoring in beach volleyball. The lack of a reliable automatic monitoring system impedes investigations about the risk factors of overuse injuries. In this work, we present an unobtrusive automatic monitoring system for beach volleyball based on wearable sensors. We investigate the possibilities of Deep Learning in this context by designing a Deep Convolutional Neural Network for sensor-based activity classification. The performance of this new approach is compared to five common classification algorithms. With our Deep Convolutional Neural Network, we achieve a classification accuracy of 83.2%, thereby outperforming the other classification algorithms by 16.0%. Our results show that detailed player monitoring in beach volleyball using wearable sensors is feasible. The substantial performance margin between established methods and our Deep Neural Network indicates that Deep Learning has the potential to extend the boundaries of sensor-based activity recognition.

Keywords Deep Learning · Convolutional Neural Network · Activity classification · Wearable sensors · Sports

Responsible editor: Johannes Fuernkranz.

✉ Thomas Kautz
thomas.kautz@fau.de

1 Digital Sports Group, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Martensstraße 3, 91058 Erlangen, Germany

2 Zentrum für Hochschulsport (ZfH), Universität Leipzig, Jahnallee 59, 04109 Leipzig, Germany

3 Immerwahrstraße 2a, 91058 Erlangen, Germany

Published online: 25 February 2017

1 Introduction

1.1 Motivation

Most injuries in volleyball and beach volleyball are caused by overuse (Aagaard et al. 1997). Shoulders, knees and the lower back are most prone to injuries due to the large loads that occur for example during a serve jump. The resulting overuse injuries are a major cause of impairment and reduced performance for professional beach volleyball players (Bahr and Reeser 2003). Also in other sports, overuse injuries play an important role, even for non-professional athletes (Covassin et al. 2012). In order to prevent such injuries, their causes need to be investigated (van Mechelen et al. 1992). Due to the multifactorial nature of sports injuries, the consideration of as many relevant risk factors as possible is necessary (Bahr and Holme 2003). This requires a fine-grained monitoring of player actions during training and matches.

By determining a profile of high load actions, training intensity and/or volume can be adapted individually and precisely for each player. This reduces overload and allows better tissue recovery, thus reducing the risk of overuse injuries (Reeser et al. 2006). For example, if a beach volleyball monitoring system reveals that a player recently performed many actions that were accompanied with high loads for the shoulder (such as spikes or overhead serves), the player could shift to actions that are less straining for the shoulder. The training could also be adapted according to tactical considerations. If the recorded activity information indicates that a player performed a lot of defensive actions but only few attacks in the current training session, the coach could adjust the training to achieve a well-balanced skill set.

However, acquiring the necessary statistics about the performed actions currently requires cumbersome manual video analysis. Although preprocessing of the videos can help extracting relevant sequences (Gomez et al. 2012; Link et al. 2010), the video analysis of a single session for a single player can take several hours. This great effort is unacceptable, especially for a consistent long-term monitoring of athletes.

An automatic detection and classification system for beach volleyball actions would facilitate the acquisition of detailed player-specific information considerably. By automatically obtaining statistics about the number and type of actions that were performed by players, it could yield crucial information for characterizing risk factors and mechanisms for overuse injuries and possibly even help to prevent them.

Wearable sensors provide a cheap and unobtrusive possibility for measuring signals that are related to player movements. Powerful pattern recognition techniques such as Deep Learning could be used for analyzing these signals. By combining wearable sensors and pattern recognition, an automatic activity recognition system for player monitoring in beach volleyball could be established.

1.2 Related work

Sensor-based activity recognition in volleyball or beach volleyball is not covered extensively in the literature. Jarning et al. (2015) attempted to determine the jump frequency in volleyball in order to understand and prevent patellar tendinopathy (also known as jumper's knee). Using accelerometer sensors, they analyzed peak vertical acceleration and peak resultant acceleration. They concluded that differences in these parameters are not sufficient for distinguishing between jumping and non-jumping movements.

Rawashdeh et al. (2015) used inertial-magnetic measurement units (IMMUs) attached to the upper arm to capture sensor signals related to the arm motion. With the acquired data, they were able to automatically distinguish between volleyball serves and baseball throws. Cuspinera et al. (2016) used gyroscopes and acceleration sensors on the wrist and hand of beach volleyball players to distinguish between four different serve types. They concluded that such a differentiation may be possible. However, their work did not include a comprehensive evaluation and results were obtained with data containing only one serve type.

IMMUs have also been used for activity recognition in various other sports, e.g., in soccer (Schuldhaus et al. 2015), skateboarding (Groh et al. 2015, 2016), snowboarding (Holleczek et al. 2010), rugby (Kautz et al. 2015), basketball (Nguyen et al. 2015), hockey (Mitchell et al. 2013) and table tennis (Blank et al. 2015). They have also been employed for the recognition of daily life activities (Bao and Intille 2004; Leutheuser e tal. 2013; Ravi et al. 2005), gesture recognition (Nguyen-Dinh et al. 2012; Roggen et al. 2015) and for activity tracking in car manufacturing (Stiefmeier et al. 2008).

In most of the aforementioned publications, classification was based on a set of generic features that were calculated from the sensor data. These features were the input to a classifier that was trained to discriminate between different classes. Typical classification algorithms were Naïve Bayes (NB, Lewis 1998), Support Vector Machine (SVM, Cortes and Vapnik 1995), k-Nearest-Neighbor (kNN, Cover and Hart 1967), decision trees (Breiman et al. 1984) and Random Forest (RF, Breiman 2001). In some cases, the classification results were improved by using voting techniques such as plurality voting (VOTE, Ho et al. 1994), boosting (Meir and Rätsch 2003) and bagging (Breiman 1996). The most successful classification algorithms in the listed publications regarding sensor-based activity recognition were SVM, kNN, Naïve Bayes, Random Forest, decision trees and a plurality voting of different base classifiers.

Deep Learning represents an alternative to these classical classification approach based on hand-crafted features. The most important advantage of Deep Learning is the automatic extraction of features, also known as representation learning (LeCun et al. 2015). Despite its successes in image and speech recognition (Deng et al. 2013; He et al. 2015; Hinton et al. 2012a), Deep Learning has not yet fully reached the field of sensor-based activity recognition (Lane and Georgiev 2015). Zheng et al. (2014) used Deep Convolutional Neural Networks (DCNNs) to classify four daily activities from accelerometer data and showed that this approach outperformed a 1-Nearest-Neighbor classifier. Zeng et al. (2014) demonstrated the use of DCNNs for human activity recognition from tri-axial accelerometers. The tested data sets included different daily life activities and activities of assembly-line workers. The authors showed that DCNNs outperformed various other methods for feature extraction, but a comparison of the DCNN with other classifiers was not included. Bailador et al. (2007) described the use of Continuous Time Recurrent Neural Networks for gesture recognition. Here, eight different gestures were classified based on the data of a tri-axial accelerometer that was held in one hand. Again, a comparison of the presented approach with other methods was not part of the study. Ordóñez and Roggen (2016) used a recurrent network architecture for gesture recognition from IMMU data. With their Deep Convolution Long-short-term Memory Neural Network, they were able to outperform various other approaches in terms of classification performance. As input, measurements from at least 10 tri-axial sensors at different body positions were required. However, this high degree of instrumentation is not acceptable in a sports context since it would hinder the mobility of athletes.

The usability of DCNNs for sports activity recognition remains to be investigated. Moreover, the performance of DCNNs in sensor-based activity recognition has not been compared systematically to other state-of-the-art activity recognition approaches. The purpose of this paper is to describe an automatic system for activity monitoring in beach volleyball. To this end, we explore the applicability of Deep Learning for sensor-based activity classification and compare it to other state-of-the-art classification methods.

2 Methods

In this section, the pipeline for the automatic activity recognition in beach volleyball is described. First, the employed hardware is specified and the conducted study is outlined. Second, a method for detecting relevant events from the raw data stream is presented. Third, different approaches for the classification of the detected events are described. At the end of this section, the evaluation procedure is outlined.

2.1 Data acquisition

2.1.1 Sensor hardware

For data acquisition, sensor units with Bosch BMA280 tri-axis acceleration sensors were used (Bosch Sensortec 2014a). Since low power consumption is an important aspect in wearable sensing, the sensors were configured to operate in low-power mode. In this configuration, the sensors consumed only $6.5\,\mu$A. In turn, the sampling rate was reduced to approximately 39 Hz.[1] Acceleration was recorded in a range of ± 16 g with a resolution of 14 bit per axis. The sensor unit also included a STM32L151CCT6 (STMicroelectronics 2015) ultra-low-power micro-controller.

2.1.2 Study design

Three-axial acceleration data was recorded from 30 subjects (11 female, 19 male) during beach volleyball training. The skills of the players ranged from beginner to

[1] According to the sensor data sheet (Bosch Sensortec 2014a), the sampling rate of the acceleration sensors in low-power mode is 40 Hz. However, our tests showed, that the actual sampling rate was only approximately 39 Hz.

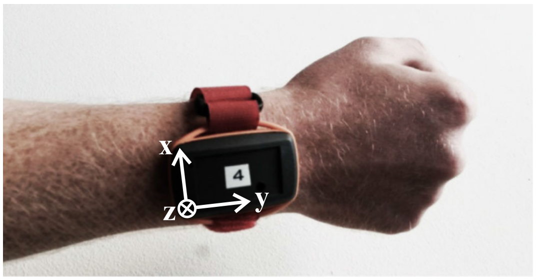

Fig. 1 Sensor attachment at the wrist of the dominant hand with a soft, thin wristband

professional level. All subjects were right-handed. The sensors were attached to the wrist of the dominant hand of the players with a soft, thin wristband. The players reported no impairment of their actions by the sensing device. The sensor attachment is depicted in Fig. 1.

The data acquisition was video recorded for reference using a GoPro Hero 3 action camera. All performed beach volleyball actions were manually labeled based on the recorded videos. The actions were grouped into the following ten classes:

1. Underhand Serve (US)
2. Overhead Serve (OS)
3. Jump Serve (JS)
4. Underarm Set (UT)
5. Overhead Set (OT)
6. Shot Attack (SA)
7. Spike (SP)
8. Block (BL)
9. Dig (DG)
10. Null Class (NL).

A short description for each of these classes is given in Table 1. A screenshot from the reference video including the measured acceleration signals is shown in Fig. 2. A summary of the recorded data is given in Table 2.

2.2 Activity detection

The first step in the data processing was the detection of potentially relevant events in the acceleration data stream. Since the detection should be executable directly on the micro-controller in the sensor device, a computationally efficient approach was necessary. A detailed description of the detection procedure is given in the following paragraphs.

Almost all actions that were relevant in our analysis involved a ball contact. Based on that, we used two assumptions about the signal characteristics for the detection of the considered beach volleyball actions. Our first assumption was, that impacts of the ball at the wrist or hand create high-frequency peaks in the acceleration data. The second assumption was, that most relevant actions are accompanied by a swing

Table 1 Description of the 10 beach volleyball action classes used in this work

Action Class	Description
Underhand Serve (US)	Underhand technique serve with different hand postures One hand is holding ball on knee/hip height and ball is hit with the other hand
Overhead Serve (OS)	Overhead technique serve w/o jumping, topspin or float serve style Ball is thrown overhead and hit with one hand
Jump Serve (JS)	Jump serve using topspin technique Ball is thrown overhead and hit while jumping
Underarm Set (UT)	Underarm set played with the forearm Ball is played on knee to hip height with both arms extended
Overhead Set (OT)	Overhead set played with the hands Ball is softly played overhead with the fingers
Shot Attack (SA)	Precise attack with sub-maximal speed Ball is hit overhead while jumping; spiking with arcing trajectory or soft dinking
Spike (SP)	Hitting ball hard with maximal speed Ball is hit overhead while jumping; spiking with open hand and downward trajectory
Block (BL)	Attempt or actual execution of blocking an opponents attack Involves a maximal jump with arms extended overhead forming a plane
Dig (DG)	Defensive play to prevent ball from hitting the ground Ball is played with either one or both extended arms while diving to the ground
Null Class (NL)	Non-beach-volleyball actions that occur between rallies Different motions like clapping, cheering, catching a ball and undefined motions

movement of the dominant arm to gain momentum and/ or to bring the arm in the correct position.

For the detection of the acceleration peaks associated with the ball impact, the acceleration signals from all three axes were high-pass filtered with a Butterworth filter with a cut-off frequency of 8 Hz. Then, the L^1-norm of the high-passed signal was computed. Subsequently, this signal was smoothed using a Butterworth low-pass filter with a cut-off frequency of 3 Hz.

If this auxiliary signal exceeded a threshold ϵ_{impact}, the next local maximum in the signal was sought. The amplitude of this maximum acceleration peak was used as an impact indicator. Since the use of the impact indicator alone was not sufficient to reliably detect relevant events, an indicator for the swing movement before the ball contact was calculated in addition. This was done by averaging the absolute acceleration values of all three axes over an interval of 200 ms before the maximum.

With the two detection features, i.e., the impact indicator and the swing indicator, a decision tree (C4.5) was employed to further refine the activity detection. A decision tree only requires threshold comparisons to perform a class prediction. Therefore, it is computationally efficient and suitable for an embedded implementation. The decision tree was trained to differentiate between relevant actions (classes

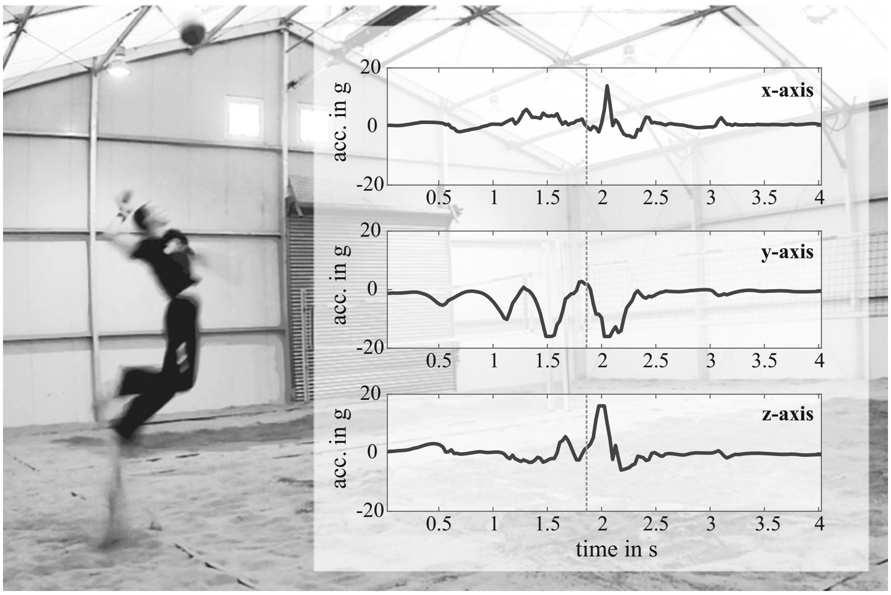

Fig. 2 Screenshot from the recorded reference video including the corresponding acceleration signals. The performed action is a jump serve. The time of the depicted frame is indicated by the *dashed lines* in the signal plots

Table 2 Summary of the recorded data

Actions of interest (without Null Class)	4284
Total duration	21 h
Subjects (female/ male)	30 (11/19)
Skill level	Beginner to professional
Sensor	BMA280 3D acceleration sensor
Sampling rate	39 Hz
Sensor placement	Wrist of dominant hand
Reference	Manually labeled video

1–9) and irrelevant actions (class 10), i.e., a binary classification problem had to be solved.

In the activity detection, it was more important to detect as many relevant events as possible (high recall) than to reject as many irrelevant events as possible (high precision). If a relevant event remained undetected (false negative), this error could not be recovered in the further processing pipeline. However, if an event was falsely assessed to be relevant by the detection algorithm (false positive) this error could be remedied in the subsequent activity classification. To implement this requirement, we designed the detection algorithm to reach a recall of 99%. This was achieved by training the decision tree in conjunction with the MetaCost algorithm (Domingos 1999) for cost sensitive learning. The cost ratio between false negatives and false positives was varied in a leave-one-subject-out-validated grid search. The cost ratio that corresponded to

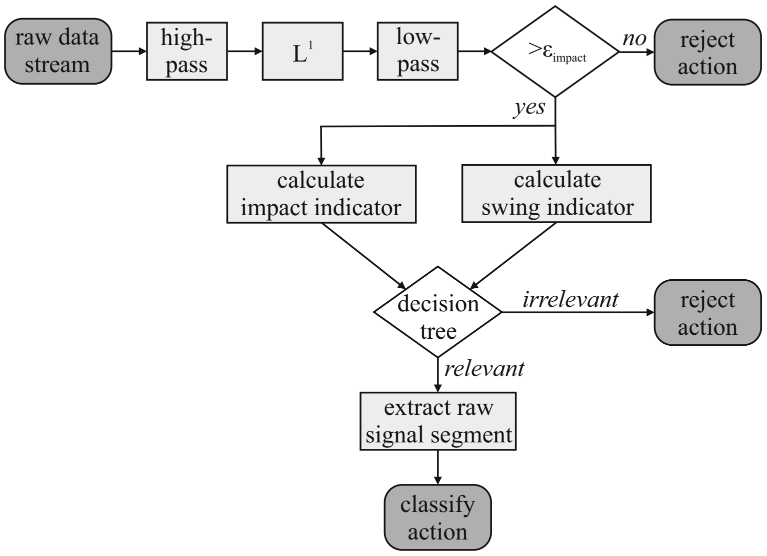

Fig. 3 Overview of the activity detection. Possibly relevant actions were extracted from the raw data stream for subsequent classification

a recall of 99% was selected and used for the training of the detection decision tree. The threshold ϵ_{impact} was defined as the minimum of the acceleration peaks of the detected relevant events in the training data.

If a potentially relevant event was detected, the raw acceleration signals from 2 s before the acceleration maximum until 2 s after the acceleration maximum (157 samples per channel) were stored for further processing. All events that were detected in the test data (true positives and false positives) in each fold of the leave-one-subject-out cross-validation were used as input for the following classification procedure. An overview of the activity detection is given in Fig. 3.

2.3 Activity classification

For the assignment of the detected actions to the ten pre-defined classes, two different approaches were investigated. The first approach included generic features in combination with different classification algorithms. In the second approach, a DCNN was used for simultaneous feature-extraction and classification. These two approaches are described in detail in the following sections.

2.3.1 Classifiers with generic features

The first approach that was employed for the classification of the detected actions was similar to the methods described in previous publications (e.g., Bao and Intille 2004; Blank et al. 2015; Leutheuser e tal. 2013; Nguyen et al. 2015; Ravi et al. 2005). The first step was the calculation of a set of manually defined features from the raw data.

Features were calculated from each of the three accelerometer axes for the detected event windows. The following features were calculated for each sensor axis:

– median
– mean
– standard deviation
– skewness
– kurtosis
– dominant frequency
– amplitude of spectrum at dominant frequency
– maximum
– minimum
– position of the maximum
– position of the minimum
– energy.

In addition, the following Pearson correlation coefficients were calculated:

– correlation between x-axis and y-axis
– correlation between x-axis and z-axis
– correlation between y-axis and z-axis.

In total, 39 features were calculated. To account for different magnitudes, the features were normalized. This was accomplished by subtracting the median from the distribution of each feature and by scaling the features according to their respective interquartile ranges. We conducted the classification with and without feature selection. The feature selection was performed with a filter approach based on the adjusted rand index (ARI, Santos and Embrechts 2009). In the classification with feature selection, only features for which the ARI was considerably higher than for the rest were used. A threshold ϵ_{ARI} was defined based on a histogram of the ARI values of the 39 features.

The class distribution of the detected events was highly imbalanced. Class imbalance can deteriorate the performance of classifiers and distort classification performance metrics (Kotsiantis et al. 2006). To avoid these negative effects, resampling was used to obtain equal instance counts for all classes in the training data. This was achieved by synthetically creating new instances from minority classes via SMOTE (Chawla et al. 2002). The resampling was performed in feature-space. After the resampling, all classes contained the same number of instances, i.e., as many instances as were originally contained in the class with the highest number of instances. Resampling was conducted after splitting the data into test and training sets to make sure that the synthetically created training samples were not influenced by the samples in the test set. The normalized and resampled feature vectors were then used as input for different classification algorithms in order to distinguish the action classes.

We tested several classifiers that were successfully applied in previous sensor-based activity classification studies (see Sect. 1.2). The tested classifiers were SVM (with radial basis function kernel), kNN, Gaussian NB, Decision Tree (CART), RF (ten trees) and a VOTE classifier based on the five aforementioned base classifiers. The optimal parameters for the SVM (parameters C and γ) and for the kNN classifier (parameter k) were determined in a grid search with $C \in \{10^0, 10^1, \ldots, 10^{10}\}$, $\gamma \in \{10^{-9}, 10^{-8}, \ldots, 10^{-2}\}$ and $k \in \{1, 2, \ldots, 10\}$.

In the voting approach, the predicted class c for an event x was obtained as the weighted vote of the predictions c_i from the $V = 5$ base classifiers:

$$c(x) = \operatorname*{argmax}_c \sum_{i=1}^{V} w_i \cdot c_i(x). \tag{1}$$

The weights w_i of the base classifiers were the classification accuracies of the corresponding classifiers on the training data. Thus, the class prediction of a classifier with a good classification accuracy was given a higher weight than the prediction of a classifier that performed poorly.

The implementation of the algorithms described in this section was based on the Python package scikit-learn (Pedregosa et al. 2011).

2.3.2 *Deep Convolutional Neural Network (DCNN)*

Background In the second approach that was tested for classification of the detected events, a DCNN was employed. Hinton et al. (2012a) defined DNNs as follows:

> A deep neural network (DNN) is a feed-forward, artificial neural network that has more than one layer of hidden units between its inputs and its outputs.

Depending on the problem to be tackled, different types of these layers, such as convolutional layers, pooling layers or fully-connected layers can be used. Convolutional layers have been found to be well suited for learning features from the raw data (Chatfield et al. 2014). In a convolutional layer, the raw data is convolved with different kernels, which can be interpreted as filtering. The weights of the filter kernels are learned from the data, i.e., the network learns specialized filters for the task at hand. A DNN containing several convolutional layers is a DCNN. By combining several convolutional layers, features with an increasing abstraction level can be learned from the data (LeCun et al. 1990).

The output of convolutional layers is usually transformed non-linearly, for example with rectified linear units (ReLUs) (Zeiler et al. 2013). The use of ReLUs in combination with convolutional layers has been found to significantly improve the performance of recognition systems (Jarrett et al. 2009). Moreover, pooling layers can help making the features learned in the convolutional layers more robust towards shifts or distortions of the input data (Boureau et al. 2010).

Fully-connected layers can be employed to find combinations of the extracted features that are suitable for classification. By combining different layers with different functions, DCNNs are able to simultaneously extract features from the raw data and perform classification based on these features.

In contrast to the classification approach that relied on generic features, the DCNN employed in this work learned features automatically. As input to the DCNN, the raw acceleration data of the detected events was employed. The input of the network was neither scaled, centered nor pre-filtered. The network input had 471 dimensions (three-axial acceleration, approximately 4 s per event, sampling rate 39 Hz).

Architecture Our DCNN consisted of an input layer, two convolutional layers with ReLUs and max-pooling, two fully-connected layers and a soft-max (Bridle 1990) output layer for classification.

A schematic representation of the DCNN designed in this work is shown in Fig. 4. The elements shown in the visualization are referenced as italic notes in the following description of the network.

The first hidden layer in our DCNN was a convolutional layer (*convolution I*). In this layer, we applied 8 kernels for filtering each of the 3 accelerometer signals. Here, the kernels were different for each axis. The kernels were one-dimensional and had a length of 32 samples (approximately 0.8 s). A bias was added to the output of each filter. The bias terms were also learned from the data and were different for each filter.

Subsequently, the filtered output of the first convolutional layer was transformed non-linearly using ReLUs (*rectification I*). The activation function f_{ReLU} of a ReLU given an input x is

$$f_{ReLU}(x) = max(0, x). \tag{2}$$

The output of the ReLUs was then compressed using temporal max-pooling (*max-pooling I*). In the max-pooling step, the output of the ReLUs within a moving pooling window were replaced by their respective maximum. For the output of the first convolutional layer, we used overlapping pooling windows with a length of 4 samples (approximately 0.1 s) and a stride of 2 samples (approximately 0.05 s).

After the temporal max-pooling, we applied a second convolutional layer (*convolution II*). This layer took the non-linearly transformed and pooled output of the first convolutional layer as input. In the second convolutional layer, we employed 16 one-dimensional filter kernels with a length of 16 samples and an additional bias for each kernel. The output of this layer was again transformed and compressed by applying the ReLU non-linearity (*rectification II*) and overlapping temporal max-pooling (*max-pooling II*, window size 6 samples, stride 4 samples).

Following the two convolutional layers, *fully-connected layers I and II* were added which took the non-linearly transformed and pooled output of the second convolutional layer as input. In addition, a scalar bias was added to each unit of the fully-connected layers. The fully-connected layers consisted of 64 ReLUs each. The final *output layer* was designed to have as many neurons as classes, i.e., the output layer consisted of 10 U in our case. As in the fully-connected layers, a bias was added to the input of each unit. For the output layer, the soft-max activation function was employed. The soft-max function f_{sm} for the c-th output unit with respect to an input vector x_c is given by

$$f_{sm}(x_c) = y_c = \frac{e^{x_c}}{\sum_{j=1}^{M} e^{x_j}} \tag{3}$$

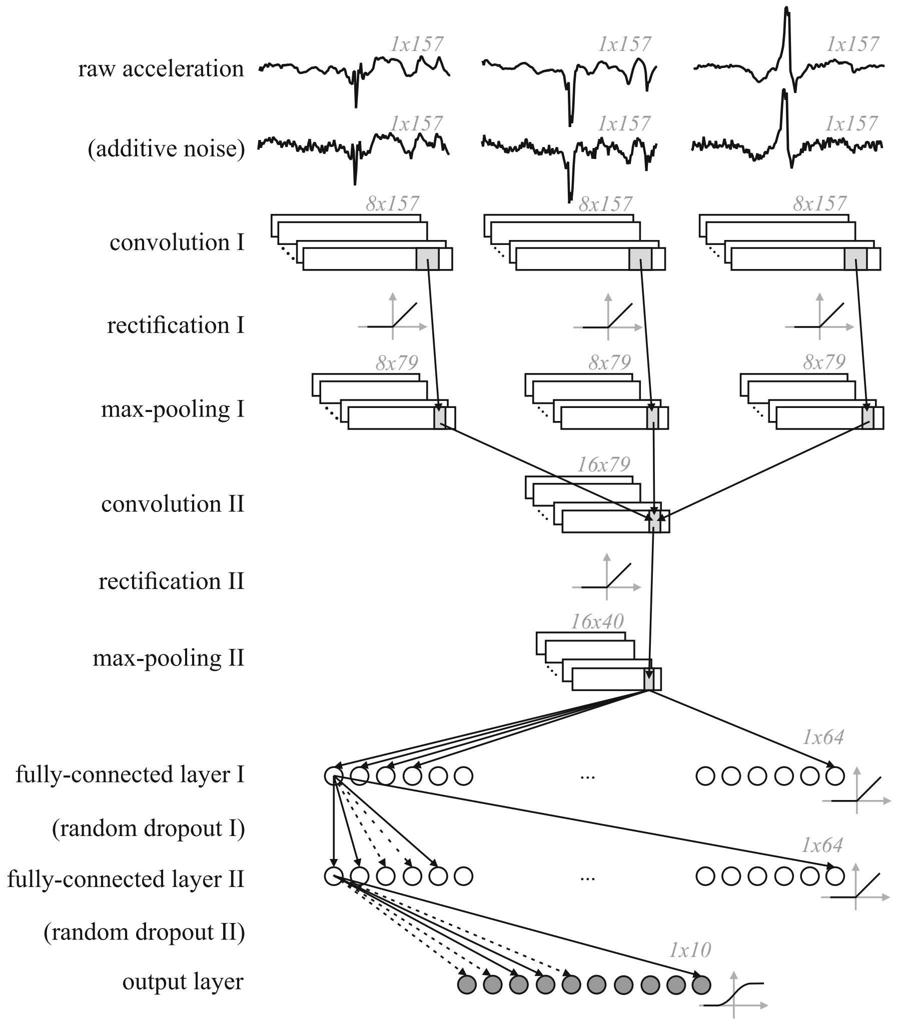

Fig. 4 Overview of the DCNN architecture. Elements in *brackets* are only active during training and not during testing. The signal dimension of the output of each layer is given in *italics*. A detailed description of the network is given in the text in Sect. 2.3.2. The example signals show the three-axial acceleration during a float serve

where $M = 10$ is the number of classes. Using this function, the activation y_c can be interpreted as an approximation of the posterior probability of class c given the input x_c (Bishop 2006), i.e.,

$$p(c|x_c) = y_c. \tag{4}$$

The class predicted by the network was defined as the one which corresponded to the output unit with the highest activation and thus exhibited the highest posterior probability.

Training A resampling procedure as described in Sect. 2.3.1 was used prior to the training of the DCNN to obtain a balanced class distribution. The only difference to the previously described procedure was, that this time the resampling was performed directly on the raw data.

The initial weights of the DCNN were drawn from a truncated normal distribution with zero mean and a SD of 0.1. All biases were initialized as 0.1. The training of the DCNN was based on the cross entropy between the output of the network $\mathbf{y}_{net}$ and the true labels $\mathbf{y}_{true}$. The cross entropy is equivalent to the log likelihood under a multinomial logit model (Jordan 1994). For the true labels, the one-hot representation was used, i.e., the label for an element from class k was a vector with M elements where $y_{true,c} = 0$, $\forall c \neq k$ and $y_{true,k} = 1$.

The vectors $\mathbf{y}_{net}$ and $\mathbf{y}_{true}$ represent discrete distributions. The c-th vector elements $y_{net,c}$ and $y_{true,c}$ give the predicted and true posterior probabilities of the current network input belonging to class c. The cross entropy H for these distributions is given by (Shore and Johnson 1981):

$$H(\mathbf{y}_{net}, \mathbf{y}_{true}) = -\sum_{c=1}^{M} y_{true,c} \cdot log(y_{net,c}). \tag{5}$$

For the training of the network, the training data was randomly split into mini-batches containing 200 detected events each. To improve the robustness and generalization abilities of the DCNN, zero-mean Gaussian noise (*additive noise*) was added to the input signals during the training process. The SD of the noise was set to 4.5 m/s^2. A different noise vector was created each time a sample was used, thus the network was never presented exactly the same input signal more than once during the training. During testing, no artificial noise was added to the data.

In each training step, the average cross entropy of all mini-batch elements was minimized. Since for each training step only the elements of the current mini-batch instead of the complete training set were evaluated, the objective function (i.e., the average cross entropy) was stochastic. For data sets with many elements, this stochastic training regime is more efficient than evaluating the objective function for the complete training data in each step (Bottou 2010).

The minimization was conducted using the Adam algorithm (Kingma and Ba 2015), a gradient-based optimizer for stochastic objective functions. The learning rate was set to 10^{-3}. The other hyperparameters for the optimization were set to $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, as suggested by Kingma and Ba (2015). To avoid overfitting, early stopping was used in the training procedure. The training was stopped, if the classification performance on a validation data set stopped increasing. As an additional regularization technique against overfitting, random dropout (Hinton et al. 2012b) was applied to the connections of the fully-connected layers (*random dropout I and II*). This was performed by randomly removing connections between units of these layers during the training. In each training step, only 30% of the connections were maintained in order to prevent complex co-adaptions between the units. For testing, no connections were dropped. We implemented and trained the DCNN using TensorFlow (Abadi et al. 2015).

2.4 Evaluation

2.4.1 Activity detection

The activity detection was evaluated in a leave-one-subject-out cross-validation. To characterize the detection performance, the total number of detections, the number of true positive, false positive and false negative detections as well as the precision were determined. The recall over all relevant events was defined as a design parameter (99%).

2.4.2 Activity classification

For the evaluation of the activity classification, a leave-three-subjects-out cross-validation was chosen. Since data was recorded from 30 different subjects, this resulted in 10 different splits with the data from 27 subjects for training and from 3 subjects for testing in each split. The subjects were split randomly and the splits were the same for all tested classifiers. The data in the training sets were resampled as described in Sect. 2.3.1 while only the original data without resampling was used for testing.

Based on the training sets, subsets of the generic features were selected according to the Adjusted Rand Index. The SVM, kNN, NB, CART, RF and VOTE classifiers were then evaluated using the selected features and using the complete feature set. For the SVM and kNN classifiers, the grid search for the parameter optimization was performed on the training data in an inner leave-three-subjects-out cross-validation. To reduce the computational cost, the parameter optimization for the SVM was conducted with 1000 randomly selected samples from the training data of each fold of the inner cross-validation.

For the training of the SVM, kNN, NB, CART, RF and VOTE classifiers with the determined parameters, all samples from the training data of the outer cross-validation were used. For the training of the DCNN, the data from 2 randomly selected subjects was held out in each fold for validation, i.e., for determining when to stop the training of the DCNN and to avoid overfitting. Hence, the DCNN was trained with the data from 25 subjects in each fold, while the other classifiers were trained with the data of 27 subjects in each fold. The classifiers based on generic features were trained with feature selection and without feature selection. An explicit feature selection for the DCNN was not necessary, since it learned features automatically.

To assess the performance of the different classification approaches, the confusion matrix, the sample accuracy and the balanced accuracy were calculated for each classifier. The sample accuracy λ_s is given by

$$\lambda_s = \frac{\sum_{c=1}^{M} r_c}{\sum_{c=1}^{M} N_c}, \tag{6}$$

where N_c is the number of samples from class c, r_c is the number of samples from class c that were classified correctly and M is the number of classes. The balanced accuracy λ_b is given by

$$\lambda_b = \frac{1}{M}\sum_{c=1}^{M}\frac{r_c}{N_c}. \tag{7}$$

The sample accuracy reflects the performance of a classifier under the assumption that the misclassification costs for the elements of a class correspond to the prior probability of this class, whereas balanced accuracy assumes equal misclassification costs over all classes. In addition to the classification performance measures, the average computation time for the classification of one detected event was determined. This time was divided into the time required for the feature calculation and the time for the actual class prediction. Also the time required for training each classifier was evaluated. Our experiments were run on a PC with an Intel Xeon E3-1276 CPU (3.6 GHz) and 32 GB RAM.

3 Results

3.1 Activity detection

With the described method for the detection of relevant events from the acceleration data stream, 12,180 events were detected. These included 4242 true positives and 7938 false positives. 42 relevant events were not detected (false negatives). The most false negatives (13) occurred for actions from the the block class which resulted in a recall of 89.2% for this class. The number of detected events in each class is depicted in Fig. 5. The overall recall of the detection algorithm was 99.0% The overall precision of the detection algorithm was 34.8%. The optimal threshold for the acceleration peak was $\epsilon_{impact} = 5.1\,\mathrm{m/s^2}$.

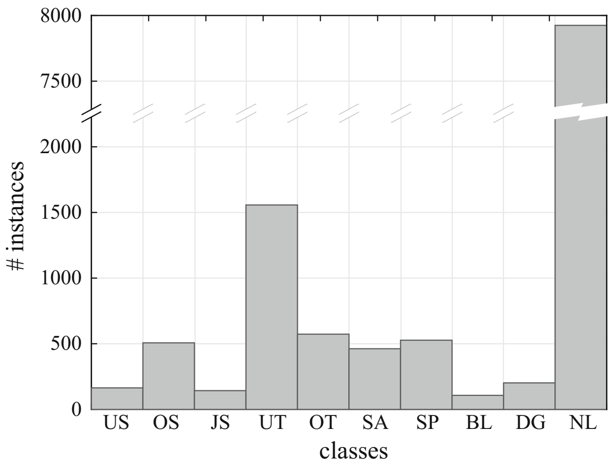

Fig. 5 Number of detected instances per class. The *vertical axis* is interrupted for better visualization due to the high number of instances in the Null Class (NL)

3.2 Activity classification

Feature selection The Adjusted Rand Index, which was used for feature selection, was low (<0.1) for all of the 39 generic features. It was positive in all cases. In the histogram of the ARI values, two groups of features were identified. For 27 of the features, the ARI was almost zero, the remaining 12 features had markedly higher ARI values. Based on this observation, we defined the threshold for the feature selection as $\epsilon_{ARI} = 5 \times 10^{-3}$. With this threshold, the same 12 features were selected in all 10 folds of the leave-three-subjects-out cross-validation. Figure 6 shows an example histogram of the ARI values of the features for one training set. The features with the highest ARI values were:

– dominant frequency a z-axis ($ARI = 0.073$)
– position of the minimum on y-axis ($ARI = 0.070$)
– position of the minimum on z-axis ($ARI = 0.058$)
– position of the maximum on x–axis ($ARI = 0.056$)

A complete list of the calculated and selected features can be found in Appendix 1. In Fig. 7, the averaged spectra of the z-axis acceleration signals for the classes Dig, Block and Jump Serve are exemplified and the dominant frequencies are indicated.

Classification accuracy The classification performance in terms of sample accuracy and balanced accuracy of the different classifiers with and without feature selection is visualized in Fig. 8 (a table with the depicted results is given in Appendix 2). The DCNN achieved the highest sample accuracy (83.2%) and the highest balanced accuracy (79.5%). The best performing classifier based on generic features was VOTE without feature selection (sample accuracy 67.2%, balanced accuracy 60.3%).

Exemplarily, the normalized confusion matrices for the DCNN, VOTE and kNN classifiers are depicted in Fig. 9. The confusion matrix for the DCNN as the best performing classifier is given in more detail in Table 3.

Computation time The average computation times for training each classifier are shown in Fig. 10. The times were averaged over the ten cross-validation folds. On average, each of the folds contained 7144 training samples after resampling. The kNN

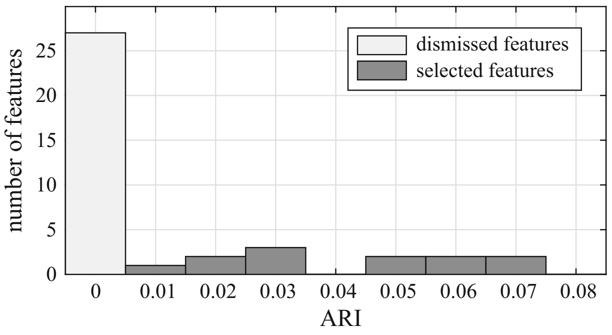

Fig. 6 Histogram of the ARI values of the 39 generic features (example for one training set)

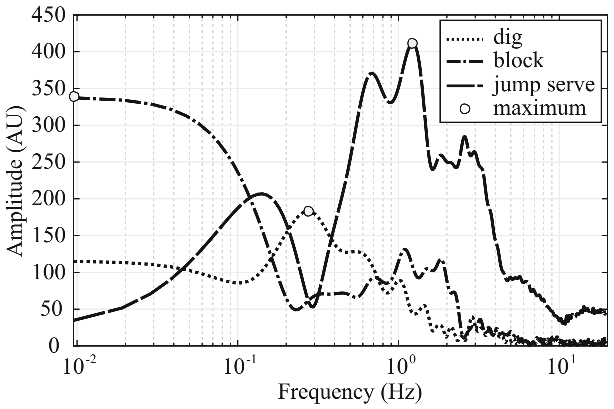

Fig. 7 Average spectra of the *z* axis acceleration signals for Digs, Blocks and Jump Serves. The *circles mark* the dominant frequencies

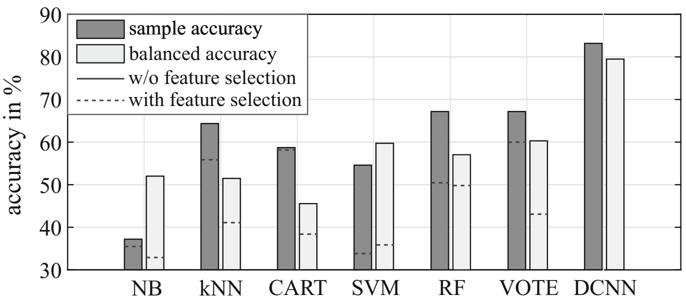

Fig. 8 Sample accuracies and balanced accuracies of the evaluated classifiers

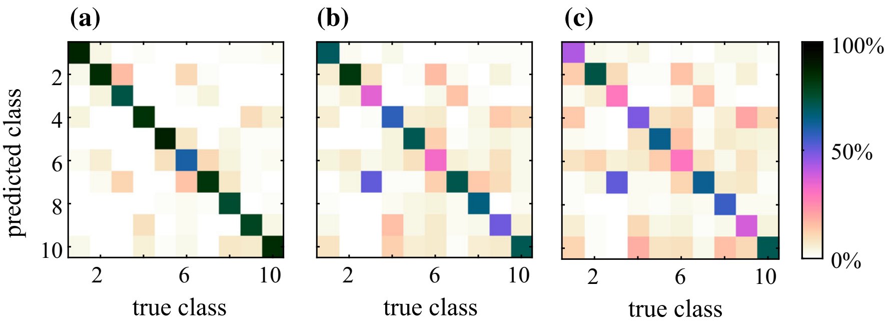

Fig. 9 Normalized confusion matrices for the **a** DCNN, **b** VOTE and **c** kNN classifiers

classifier does not require training. The computation time given for this classifier is the time that was required for building the data representation used by scikit-learn. The shortest training time was required for NB classifier with feature selection (18.1 ms). The training of the DCNN took the most time (759.9 s). Of the classifiers using generic features, the SVM (671.1 s) and VOTE (686.1 s) classifiers with feature selection had the longest training times.

Table 3 Normalized confusion matrix for the DCNN, columns represent the true class, rows represent the predicted class, results in %

	US	OS	JS	UT	OT	SA	SP	BL	DG	NL
US	**87.2**	3.7	0.0	1.4	0.0	0.0	0.0	0.9	1.0	1.9
OS	3.0	**84.6**	16.1	0.3	0.0	10.6	1.3	0.0	0.0	0.3
JS	0.0	3.9	**72.0**	0.0	0.0	0.9	4.0	0.0	0.0	0.0
UT	4.3	0.0	0.0	**83.2**	0.7	0.4	0.0	0.0	9.4	4.9
OT	0.0	0.0	0.0	0.1	**87.6**	7.1	0.6	3.7	1.5	1.1
SA	1.8	5.7	0.7	0.1	8.9	**60.2**	10.2	4.7	1.5	2.2
SP	0.0	2.0	11.2	0.1	0.2	14.1	**82.2**	7.5	1.5	0.6
BL	0.0	0.0	0.0	0.2	0.9	2.2	0.9	**75.7**	0.5	1.0
DI	0.6	0.0	0.0	9.2	0.3	1.7	0.4	0.0	**77.7**	3.6
NL	3.0	0.0	0.0	5.3	1.4	2.8	0.4	7.5	6.9	**84.5**

Details about the classes can be found in Table 1
Bold values indicate the class-wise sensitivity

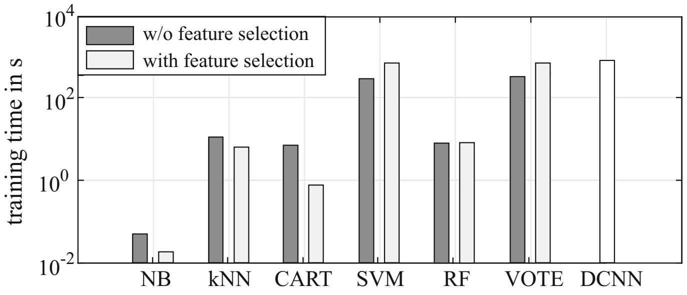

Fig. 10 Average time in seconds for training each classifier on the training data of each fold of the cross-validation (7144 samples on average)

The average computation times for performing one classification (feature calculation and class prediction) with each of the evaluated classifiers are shown in Fig. 11. The time for calculating the 39 generic features (required for the NB, kNN, CART, SVM and VOTE classifiers) for one action was 2.24 ms on average. The time for calculating only the 12 selected features was 0.59 ms on average.

The shortest computation time (0.53 μs) for the class prediction was achieved with the CART classifier (approximately same time with and without feature selection). Class prediction with the SVM with feature selection and VOTE without feature selection took the most time (2.85 and 3.64 ms). When only the 12 selected features were used, the computation time for the prediction was reduced for the NB, kNN, CART, RF and VOTE classifiers. The prediction time of the SVM increased slightly with fewer features.

One classification with combined feature calculation and class prediction using the DCNN took 0.06 ms on average. Hence, the DCNN required the shortest computation time for the complete classification.

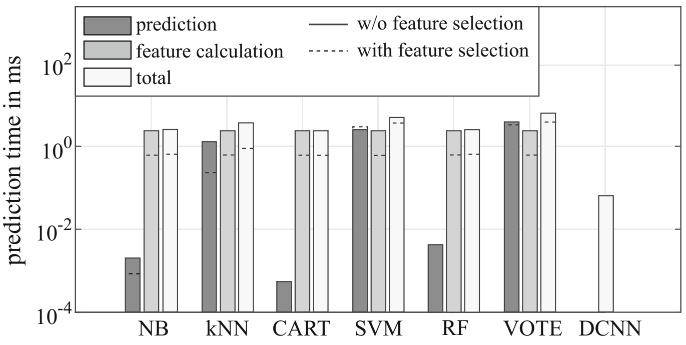

Fig. 11 Average time in milliseconds for classifying one detected event (logarithmic scale). The classification time consists of the time taken for feature calculation and the time for the actual class prediction from these features. In the DCNN, feature calculation and prediction are combined and cannot be separated in terms of computation time

4 Discussion

4.1 Activity detection

The results of the activity detection showed, that it was possible to detect potentially relevant events with a high recall of 99%. Hence, almost all relevant events were detected. The class with the highest number of false negatives was the block class. A closer examination revealed, that in all of these cases the blocks were only attempted, i.e., no ball contact occurred. However, the detection algorithms was mainly designed for actions that involve a ball contact, which explains the lower performance of the detection algorithm on attempted blocks. Nevertheless, a high class-wise recall of 89.2% was achieved even for the block class.

The relatively low overall precision of 34.8% is acceptable, since the activity detection represents only the first step in the processing pipeline and false positive detections were removed in the subsequent classification step.

4.2 Activity classification

4.2.1 Classifiers with generic features

The performance of the different approaches for the classification of the detected activities was inhomogeneous. No satisfactory results could be obtained with the NB, kNN, CART, SVM, RF, and VOTE classifiers on the presented data set (maximum sample accuracy 67.2%, maximum balanced accuracy 60.3% with VOTE). In comparable studies which were also related to activity classification using inertial sensors (see Sect. 1), high classification rates of more than 90% were achieved with the same classifiers and similar features. One possible explanation for this discrepancy is the fact that in the current study only accelerometer data was used to reduce power consumption. In other studies, also gyroscope measurements were employed. Turning rate data from gyroscopes could provide information about the rotation of the sensor which in turn could be used for the activity classification. However, gyroscopes would increase the power consumption for sensing substantially. For example the Bosch BMG160 (Bosch Sensortec 2014b), a state-of-the-art tri-axial gyroscope sensor from the same manufacturer as the employed acceleration sensors would consume 5 mA, which is 769 times the power consumption of the acceleration sensor. Apart from the additional gyroscope information, higher sampling rates above 100 Hz for the acceleration sensors were used in other studies. Also the acceleration sensor employed in our study would allow sampling rates up to 200 Hz instead of the 39 Hz that were used in the presented study. However, this would increase the power consumption of the sensor by a factor of 20. Using gyroscopes or higher sampling rates would considerably decrease battery run-time and thus reduce the usability of the sensing device.

The computation time required for the training of the classifiers that were based on generic features decreased in most cases when feature selection was applied prior to the training. In contrast, the training time of the SVM with feature selection increased. This was due to the fact, that more support vectors were required to separate the classes in the lower dimensional feature space. Since the SVM was the slowest component of the VOTE classifier, the training time for the voted classification also increased with feature selection. The RF classifier exhibited the best trade-off between computation time and classification performance among the classifiers with generic features. While achieving only 0.1% lower sample accuracy than the VOTE classifier, it required only a fraction of the time for training and prediction.

The employed classifiers all suffered from overfitting, i.e., they performed much better on the training data then on the test data. A possible remedy for this problem would be to record more training data from more players. In the case of the tree-based classifers (CART, RF), overfitting was partly caused by the depth of the trees that were fitted based on the training data (depth > 50). Pruning strategies could possibly reduce overfitting for these classifiers.

The performance of the classifiers that were based on generic features did not improve when feature selection was employed. Instead, the classification accuracy with feature selection was consistently lower compared to the experiments with the complete feature set. At the same time, computation times for feature calculation and prediction decreased in almost all cases when only 12 features were used. Only the prediction time (without feature calculation) of the SVM classifier with 12 features was higher than with 39 features. Again, this was caused by the higher number of support vectors in the lower dimensional feature space. The advantage of a slight decrease in total computation time required for the classification with fewer features can not outweigh a drop in accuracy by more than 20% as it occurred for example with the SVM. A wrapper approach for feature selection (Kohavi and John 1997) might yield better results than the presented filter approach, but in conjunction with cross-validation and parameter selection for 5 different classifiers and more than 12,000 samples, the associated computational cost is prohibitive.

Even though some of the features, as for example the dominant frequency along the z-axis, varied markedly across subsets of the classes (see Fig. 7), the ARI values were low for all features when all ten classes were considered. This indicates that the generic features did not exhibit sufficient discriminative power for the problem at hand. Hence, even selecting an optimal subset of these featured may not yield satisfactory results. Alternatively, application-specific features, derived by experts from the motion characteristics of the analyzed actions could be used for the classification. However, due to the high degree of similarity between the classified movements, an increase of classification performance with such features is questionable.

We assume that classification errors were also caused by the uncertainty of the labeling. Especially for players with a lower skill-level, it can be difficult even for an expert to assign a performed action to one of the defined classes.

4.2.2 Deep Convolutional Neural Network

The unsatisfactory results of the established methods on the presented beach volleyball data set show, that a novel approach for sensor-based activity classification is necessary. The DCNN outperformed the classifiers based on generic features by a large margin of 16.0% in terms of sample accuracy and by 19.2% in terms of balanced accuracy. This suggests that Deep Learning is a viable solution for the problem of sensor-based activity classification.

From the confusion matrix of the DCNN (see Fig. 9a; Table 3), a more detailed analysis of the classification outcome of the DCNN is possible. The action classes that were confused most often by the DCNN were mostly classes for which the corresponding movements were very similar. Amongst the actions that were confused most often were Underarm Set versus Dig (UT–DG), Overhead Serve versus Jump Serve (OS–JS), Spike versus Shot Attack (SP–SA) and Jump Serve versus Spike (JS–SP). For example, during a Jump Serve and a Spike, the movements of the dominant arm are very similar. Here, the use of a second sensor unit at the non-dominant hand could improve the classification performance, since in the case of a Jump Serve it is used to throw the ball in the air before the hit which does not occur in a Spike. We assume that the similar actions that were confused were also similar in terms of the associated load on the joints in most cases. Hence, the confusion of these similar actions in the context of overuse injury analysis is acceptable. However, the misclassifications reduce the reliability of tactical considerations that are based on the classified actions.

In addition to the good classification accuracy, the computational cost of the DCNN also compared favorably to the other evaluated classifiers. The average computation time that was required for classifying one detected event with our DCNN was much shorter than for all classifiers based on generic features. For example, the computation time for the SVM with 12 features was increased by a factor of approximately 75 compared to the DCNN. Although the class prediction itself could be performed in very little time, for example with NB and CART classifiers, the computational overhead for the explicit calculation of the generic features eliminated this advantage. In contrast, the DCNN performed a combined feature calculation and class prediction with low computational cost. The training time for our DCNN was similar to that for the best-performing classifier based on generic features (VOTE). These results show, that both good classification performance and computational efficiency can be achieved at the same time with the proposed DCNN. However, the computation time depends on the concrete implementation and used hardware, therefore the generalizability of these results is limited.

Another major advantage of the DCNN is its ability to automatically extract features from the raw data. Hence, no cumbersome feature design, feature selection or other pre-processing of the acceleration data of the detected events were necessary.

Although the fundamental concepts of the DCNN are applicable for a wide range of problems, the specific realization of the network architecture and the training regime needs to be adjusted in order to obtain useful results for the problem at hand. Theoretically, myriads of different network designs are possible. Some of the design parameters for the architecture are for example the number, type, size and sequence of network layers, the way of connecting subsequent layers as well as the initialization and activation functions for the involved units. Also for the training, various parameters need to be defined, e.g., the optimization algorithm and its hyperparameters, the objective function, possibilities for regularization and stopping criteria.

The large number of parameters makes it computationally unfeasible to find the optimal architecture settings in an automatic manner. Therefore, the network architecture and training scheme presented in this work had to be determined empirically. Fortunately, research by Hinton et al. (2012a) suggests, that the performance of Deep Neural Networks is relatively insensitive to the precise details of the network architecture. In our tests, we found that the use of additional layers did not improve the classification performance. Also a change in the number of units in each layer did not alter the results considerably. However, our DCNN suffered from overfitting, i.e., its prediction performance was much higher on the training data than on the validation and test data. This effect was reduced by the pooling layers, by adding artificial noise to the training data and by applying random dropout to the connections of the fully connected layers. The use of random dropout on the input data did not improve the performance, probably because this had a similar effect as adding artificial noise to the input. Overfitting might be further reduced by collecting more data for training and by unsupervised pre-training of the network. Moreover, training of the network with batch normalization (Ioffe and Szegedy 2015) could possibly help avoid overfitting.

4.3 System implementation

The sensor unit in our study was optimized for low power-consumption in order to allow a long-term monitoring of player actions. Since the proposed activity detection algorithms requires only little pre-processing of the raw data and several threshold comparisons in a decision tree, it is suitable for embedded implementation in the sensor unit. The presented classification methods were not yet optimized for computational efficiency and embedded implementation. However, modern mobile computing platforms (e.g., Qualcomm Zeroth, Contreras 2015) are optimized especially for the execution of Deep Learning algorithms like the presented DCNN classification. Currently, missing mobile implementation of the classification is remedied by storing the data for the detected events in the sensor unit and performing off-line classification of the actions on an external device.

In our study, all subjects were right-handed. Left-handed players would have to wear the sensor unit at their left wrist. In order to account for this different attachment, the signals from the sensor axis that is parallel to forearm (y-axis) would need to to be inverted. With this correction, we presume that the same detection and classification algorithms as for right-handed players could be used.

Although the proposed system was developed and evaluated in the context of beach volleyball, it might also be used for volleyball. Volleyball involves very similar movements as beach volleyball and also the injuries associated to both types of sport are comparable. However, the application of the proposed system was not tested in a volleyball scenario.

5 Summary and conclusion

In this paper, a novel monitoring system for beach volleyball was described. Using data from wrist-worn acceleration sensors, 10 different types of player actions that occur in beach volleyball were detected and classified. For the classification, we compared 5 different classification algorithms that were successfully used in similar studies to a Deep Learning approach. Adequate classification results could only be achieved with the proposed DCNN. It outperformed the second-best classifier by 16.0% in terms of sample accuracy and by 19.2% in terms of balanced accuracy while at the same time requiring less computation time for classification on the same hardware.

The proposed sensor-based monitoring system is the first one that allows a detailed and automatic monitoring of beach volleyball actions. Thereby, our system significantly facilitates and speeds up data acquisition and analysis for studies in sports science. It allows building up extensive data sets containing fine-grained information about the actions performed by professional and non-professional beach volleyball players. In addition to the potential for training optimization, the output provided by our system in combination with medical records can be used to asses the risk factors of overuse injuries related to beach volleyball.

Moreover, we provide the first systematic comparison of various widely-used classifiers based on generic features with a DCNN in the context of sensor-based movement classification. The good performance of the DCNN compared to previously reported methods indicates that the application of Deep Learning for sensor-based movement classification is a promising research field and has the potential for achieving higher levels of classification accuracy.

In future, we will explore possibilities for implementing the complete data processing pipeline including the classification in a wearable system to provide results in real-time and on the wearable device. Furthermore, options for replacing the heuristic action detection algorithm by a computationally efficient Deep Learning approach will be investigated.

Acknowledgements We would like to thank Markus N. Streicher, Alexander Stolpe, Christoph Eckelmann, Adib Taraben and David Reck for their valuable suggestions and their contributions to the development of the sensor hardware as well as the recording and labeling of the data. Bjoern Eskofier gratefully acknowledges the support of the German Research Foundation (DFG) within the framework of the Heisenberg professorship programme (grant number ES 434/8-1).

Appendix 1: Generic features

The following generic features were calculated for the classification with the NB, kNN, CART, SVM, RF and VOTE classifiers. The features with the highest ARI values that were used for the experiments with feature selection are printed in bold type.

1. standard deviation x-axis
2. standard deviation y-axis
3. standard deviation z-axis
4. mean x-axis
5. mean y-axis
6. mean z-axis
7. median x-axis
8. median y-axis
9. median z-axis
10. skewness x-axis
11. skewness y-axis
12. skewness z-axis
13. kurtosis x-axis
14. kurtosis y-axis
15. kurtosis z-axis
16. **dominant frequency x-axis**
17. dominant frequency y-axis
18. **dominant frequency z-axis**
19. amplitude of spectrum at dominant frequency x-axis
20. amplitude of spectrum at dominant frequency y-axis
21. amplitude of spectrum at dominant frequency z-axis
22. minimum x-axis
23. **minimum y-axis**
24. **minimum z-axis**
25. maximum x-axis
26. **maximum y-axis**
27. **maximum z-axis**
28. **position of the maximum x-axis**
29. **position of the maximum y-axis**
30. **position of the maximum z-axis**
31. **position of the minimum x-axis**
32. **position of the minimum y-axis**
33. **position of the minimum z-axis**
34. energy x-axis
35. energy y-axis
36. energy z-axis
37. correlation between x-axis and y-axis
38. correlation between x-axis and z-axis
39. correlation between y-axis and z-axis

Appendix 2: Classification accuracy

The performance of the compared classification approaches is summarized in Table 4.

Table 4 Sample accuracy (SAC) and balanced accuracy (BAC) for all compared classifiers with and without feature selection (%)

	w/o feature selection		With feature selection	
	SAC	BAC	SAC	BAC
NB	37.2	52.0	35.6	33.0
kNN	64.4	51.5	55.9	41.1
CART	58.7	45.6	58.2	38.4
SVM	54.6	59.7	33.9	35.8
RF	67.1	57.0	49.5	50.0
VOTE	67.2	60.3	49.9	56.0
DCNN	**83.2**	**79.5**	**83.2**	**79.5**

The best value in each column is given in bold type

References

Aagaard H, Scavenius M, Jørgensen U (1997) An epidemiological analysis of the injury pattern in indoor and in beach volleyball. Int J Sports Med 18(3):217–221

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. http://download.tensorflow.org/paper/whitepaper2015.pdf. Accessed 2 Dec 2015

Bahr R, Holme I (2003) Risk factors for sports injuries—a methodological approach. Br J Sports Med 37(5):384–392

Bahr R, Reeser JC (2003) Injuries among world-class professional beach volleyball players. The Fédération Internationale de Volleyball beach volleyball injury study. Am J Sports Med 31(1):119–125

Bailador G, Roggen D, Tröster G, Trivino G (2007) Real time gesture recognition using continuous time recurrent neural networks. In: Proceedings of the international conference on body area networks

Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data. In: Proceedings of the 2nd international ICST conference on body area networks (BODYNETS-07), Florence, Italy

Bishop CM (2006) Pattern recognition and machine learning. Springer, New York

Blank P, Hoßbach J, Schuldhaus D, Eskofier BM (2015) Sensor-based stroke detection and stroke type classification in table tennis. In: Proceedings of the ACM international symposium on wearable computers, Osaka, Japan, pp 93–100

Bosch Sensortec (2014a) BMA280—digital, triaxial acceleration sensor, Data Sheet, Reutlingen, Germany. http://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMA280-DS000-11_published.pdf. Accessed 16 Nov 2015

Bosch Sensortec (2014b) BMG160—digital, triaxial gyroscope sensor, data sheet, Reutlingen, Germany. http://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMG160-DS000-09.pdf. Accessed 16 Nov 2015

Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: 19th international conference on computational statistics, Physica-Verlag, Paris, France, pp 177–186

Boureau Y, Ponce J, LeCun Y (2010) A theoretical analysis of feature pooling in visual recognition. In: Proceedings of the 27th international conference on machine learning (ICML-10)

Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton

Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Fogleman-Soulie F, Herault J (eds) Neurocomputing: algorithms, architectures and applications. Springer-Verlag Berlin, pp 227–236

Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: delving deep into convolutional nets. arXiv:1405.3531v4

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16(3):321–357

Contreras I (2015) Qualcomm research brings server-class machine learning to everyday devices-making them smarte. Technical report, Qualcomm Technologies, Inc., San Diego, CA. https://www.qualcomm.com/news/onq/2015/10/01/qualcomm-research-brings-server-class-machine-learning-everyday-devices-making. Accessed 12 Jan 2016

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297

Covassin T, Cheng G, Nayar S, Heiden E (2012) Epidemiology of overuse and acute injuries among competitive collegiate athletes. J Athl Train 47(2):198–204

Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27

Cuspinera LP, Uetsuji S, Ordonez J, Roggen D (2016) Wearable beach volleyball serve type recognition. In: Proceedings of the 20th international symposium on wearable computers

Deng L, Li J, Huang J-T, Yao K, Yu D, Seide F, Seltzer M, Zweig G, He X, Williams J, Gong Y, Acero A (2013) Recent advances in deep learning for speech research at Microsoft. In: Proceedings of the international conference on acoustics, speech and signal processing (ICASSP), Vancouver, Canada, pp 8604–8608

Domingos P (1999) MetaCost: a general method for making classifiers cost-sensitive. In: Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, USA, pp 155–164. ISBN 1581131437. doi:10.1145/312129.312220

Gomez G, Linarth A, Link D, Eskofier BM (2012) Semi-automatic tracking of beach volleyball players. In: 9. Symposium der Sektion Sportinformatik der Deutschen Vereinigung fr Sportwissenschaft, p 22

Groh BH, Kautz T, Schuldhaus D, Eskofier BM (2015) Imu-based trick classification in skateboarding. In: Proceedings of the KDD workshop on large-scale sports analytics, Sydney, Australia

Groh BH, Fleckenstein M, Eskofier B (2016) Wearable trick classification in freestyle snowboarding. In: Proceedings of the 13th annual international body sensor networks conference, San Francisco, USA, pp 89–93

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. arXiv:1502.01852

Hinton G, Deng L, Yu D, Dahl GE, Mohamed A-Rahman, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B (2012a) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97

Hinton G, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012b) Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580

Ho TK, Hull JJ, Srihari SN (1994) Decision combination in multiple classifier systems. IEEE Trans Pattern Anal Mach Intell 16(1):66–75

Holleczek T, Schoch J, Arnrich B, Troster G (2010) Recognizing turns and other snowboarding activities with a gyroscope. In: Proceedings of the international symposium on wearable computers (ISWC), Seoul, South Korea, pp 1–8

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167

Jarning JM, Mok K-M, Hansen BH, Bahr R (2015) Application of a tri-axial accelerometer to estimate jump frequency in volleyball. Sports Biomech 14(1):95–105

Jarrett K, Kavukcuoglu K, Ranzato M, LeCun Y (2009) What is the best multi-stage architecture for object recognition? In: Proceedings of the 12th international conference on computer vision, Kyoto, Japan, pp 2146–2153

Jordan MI (1994) A statistical approach to decision tree modeling. In: Proceedings of the 7th annual conference on computational learning theory

Kautz T, Groh BH, Eskofier BM (2015) Sensor fusion for multi-player activity recognition in game sports. In: Proceedings of the KDD workshop on large-scale sports analytics, Sydney, Australia

Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: Proceedings of the international conference on learning representations, San Diego, USA, pp 1–13

Kohavi R, John GH (1997) Wrappers for feature subset selection. Artif Intell 97(1):273–324

Kotsiantis S, Kanellopoulos D, Pintelas P (2006) Handling imbalanced datasets: a review. Science 30:25–36

Lane ND, Georgiev P (2015) Can deep learning revolutionize mobile sensing? In: Proceedings of the 16th international workshop on mobile computing systems and applications, Santa Fe, USA, pp 117–122

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1990) Handwritten digit recognition with a back-propagation network. In: Touretzky D (ed) Advances in neural information processing systems. Morgan Kaufmann, San Francisco, vol 2, pp 396–404

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

Leutheuser H, Schuldhaus D, Eskofier BM (2013) Hierarchical, multi-sensor based classification of daily life activities: comparison with state-of-the-art algorithms using a benchmark dataset. PLoS ONE 8(10):e75196

Lewis DD (1998) Naive bayes at forty: the independence assumption in information retrieval. In: Proceedings of the 10th European conference on machine learning, Chemnitz, Germany, pp 4–15

Link D, Haag T, Rau C, Lames M (2010) Wettkampfanalyse im Beachvolleyball mittels Positionsdaten. BISp Jahrb Forsch 11:171–174

Meir R, Rätsch G (2003) An introduction to boosting and leveraging. In: Mendelson S, Smola AJ (eds) Advanced lectures on machine learning. Springer, New York, pp 118–183

Mitchell E, Monaghan D, O'Connor NE (2013) Classification of sporting activities using smartphone accelerometers. Sensors 13(4):5317–5337

Nguyen LNN, Rodríguez-Martín D, Català A, Pérez-López C, Samà A, Cavallaro A (2015) Basketball activity recognition using wearable inertial measurement units. In: Proceedings of the 16th international conference on human computer interaction, Heraklion, Greece, p 60

Nguyen-Dinh L.-V, Roggen D, Calatroni A, Tröster G (2012) Improving online gesture recognition with template matching methods in accelerometer data. In: Proceedings of the 12th international conference on intelligent systems design and applications (ISDA), pp 831–836

Ordóñez FJ, Roggen D (2016) Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 16(1):115

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Ravi N, Dandekar N, Mysore P, Littman ML (2005) Activity recognition from accelerometer data. In: Proceedings of the 17th national conference on innovative applications of artificial intelligence, Pittsburgh, USA, vol 5, pp 1541–1546

Rawashdeh SA, Rafeldt DA, Uhl TL, Lumpp JE (2015) Wearable motion capture unit for shoulder injury prevention. In: Proceedings of the IEEE 12th international conference on wearable and implantable body sensor networks (BSN), London, UK, pp 1–6

Reeser JC, Verhagen E, Brinner WW, Askeland I, Bahr R (2006) Strategies for the prevention of volleyball related injuries. Br J Sports Med 40(7):594–600

Roggen D, Cuspinera LP, Pombo G, Ali F, Nguyen-Dinh L-V (2015) Limited-memory warping LCSS for real-time low-power pattern recognition in wireless nodes. In: Abdelzaher T, Pereira N, Tovar E (eds) Wireless sensor networks. Springer, Cham, pp 151–167

Santos JM, Embrechts M (2009) On the use of the adjusted rand index as a metric for evaluating supervised classification. In: 19th international conference on artificial neural networks (ICANN 2009), Limassol, Cyprus, pp 175–184

Schuldhaus D, Zwick C, Körger H, Dorschky E, Kirk R, Eskofier BM (2015) Inertial sensor-based approach for shot/pass classification during a soccer match. In: Proceedings of the KDD workshop on large-scale sports analytics, Sydney, Australia

Shore JE, Johnson RW (1981) Properties of cross-entropy minimization. IEEE Trans Inf Theory 27(4):472–482

Stiefmeier T, Roggen D, Ogris G, Lukowicz P, Tröster G (2008) Wearable activity tracking in car manufacturing. IEEE Pervasive Comput 2:42–50

STMicroelectronics (2015) STM32L151CC microcontroller, data sheet, Geneva, Switzerland. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00048356.pdf. Accessed 16 Nov 2015

van Mechelen W, Hynek H, Kemper HCG (1992) Incidence, severity, aetiology and prevention of sports injuries. Sports Med 14(2):82–99

Zeiler MD, Ranzato M, Monga R, Mao M, Yang K, Le QV, Nguyen P, Senior A, Vanhoucke V, Dean J, et al (2013) On rectified linear units for speech processing. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP), Vancouver, Canada, pp 3517–3521
Zeng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, Zhang J (2014) Convolutional neural networks for human activity recognition using mobile sensors. In: Proceedings of the 6th international conference on mobile computing, applications and services (MobiCASE), Austin, USA, pp 197–205
Zheng Y, Liu Q, Chen E, Ge Y, Zhao JL (2014) Time series classification using multi-channels deep convolutional neural networks. In: Li F, Li G, Hwang S, Yao B, Zhang Z (eds) Web-age information management. Springer, Cham, pp 298–310