Chapter

Application of Deep Learning Methods for Detection and Tracking of Players

Marina Ivasic-Kos, Kristina Host and Miran Pobar

Abstract

This chapter deals with the application of deep learning methods in sports scenes for the purpose of detecting and tracking the athletes and recognizing their activities. The scenes recorded during handball games and training activities will be used as an example. Handball is a team sport played with the ball with well-defined goals and rules, with a given number of players who can participate in the game as well as their roles. Athletes move quickly throughout the field during the game, change position and roles from defensive to offensive, use different techniques and actions, and very often are partially or completely occluded by another athlete. If artificial lighting and cluttered background are additionally taken into account, it is clear that these are very challenging tasks for object detectors and trackers. The chapter will present the results of various experiments that include player and ball detection using state-of-the-art deep convolutional neural networks such as YOLO v3 or Mask R-CNN, player tracking using Deep Sort, key player determination using activity measures, and action recognition using LSTM. In the conclusion, open issues and challenges in applying deep learning methods in such a dynamic sports environment will be discussed.

Keywords: handball, object detector, object tracking, action recognition, person detection, deep convolutional neural networks, YOLO, mask R-CNN, LSTM, DeepSort, Hungarian algorithm, optical flow, STIPs

1. Introduction

Computer vision (CV) is a compelling field of Artificial intelligence that develops the theory and methods by which information about the real world can be automatically extracted and analyzed from image data. Image data can be in many forms, such as image, video, depth image, multi-camera views, or multidimensional data from a medical scanner.

The objective of CV is to model the real world or to recognize objects from digital images enabling computers or devices to „see", interpret, manipulate, analyze, and understand what was seen and draw conclusions about the properties of the 3D world based on a given image or a sequence of images [1].

Basic CV tasks are image classification, segmentation, similarity calculation and object localization. Recognition of objects present on the scene and their features (e.g., shapes, textures, colors, sizes, spatial arrangement,) is often prerequisite

for more complex CV tasks such as image retrieval, image description, object detection, object tracking, action recognition, image or scene analysis, and image understanding [2].

In all tasks, the starting point are the image features that carry important information and need to be extracted and processed in order to generate new information and conclusions. The image features can be divided into low-level features such as corners, edges, or contours that can be extracted with relatively simple image operations, and high-level features that require domain-knowledge to get structured information related to the object or action being taken [3].

Feature extraction can be described as a pre-processing step to remove redundant parts from the data and keeping the key information for accomplishing the task. Some well-known features that can be extracted are Optical flow for extracting motion information, Histogram of Oriented Gradients (HOG) and Silhouette for extracting shape information, Space–Time Interested Points for extracting interest points, etc.

To accomplish typical CV tasks, Image processing and Machine learning (ML) play an important role. Image processing is focused on low-level features and manipulation of image data for normalizing photometric properties of visual data, removing digital noise, data augmentation, etc., and is not concerned with understanding the content of visual data. However, when it comes to interpret the content and draw conclusions about the image to automate CV tasks, the most important fields are ML and its subfield Deep learning (DL) [2].

Before DL, computer vison tasks required a lot of coding and manual effort to define the features that can be extracted from images, with little automation involved [4]. With DL methods such as Convolutional Neural Network (CNN) [5], much of that work related the features to be extracted can be inferred automatically from data. Even though many features can be extracted automatically in the CNN framework for different tasks, manual feature extraction can still be useful for either augmenting the automatically extracted features or perform other tasks such as temporal segmentation of video or detection of active players.

Typical CV tasks such as object detection, object tracking, and action recognition, the tasks we will focus on the most here, are supervised learning tasks (**Figure 1**). Supervised learning relies on labeled ground truth data, based on which the learning algorithm infers the mapping between the raw data and the desired labels in the training stage. Thus, a prerequisite for supervised learning is data preparation, which includes data collection and labeling, pre-processing and feature extraction, followed by splitting data into a training and testing set, then selecting an appropriate learning algorithm and model structure for the specific task. After training and validating the model, the model needs to be tested and the obtained results need to be compared with the ground-truth data to evaluate the

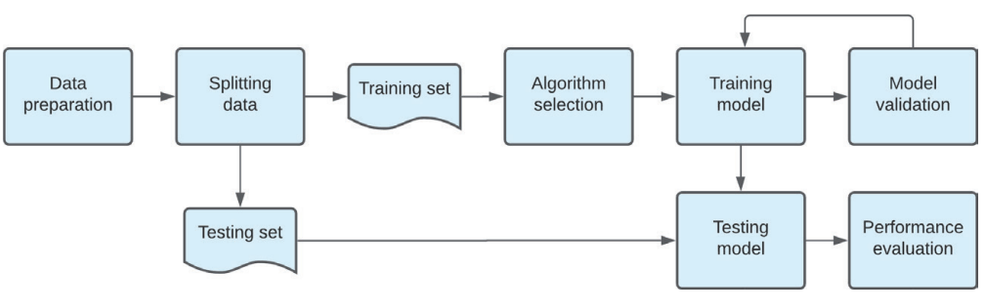

Figure 1.
Supervised learning process.

performance. The performance evaluation is represented with different metrics appropriate for the specific task.

CV tasks can be implemented for image and video analysis in different domains, including the sports domain. Various CV techniques can be very useful for all parties interested in analyzing the game, including the coach, the reporters, the referee team, the physiotherapists and others, for making decisions about an occurred event, for monitoring and comparing the performance of each individual player, for choosing a strategy, for fast automatic analysis of video materials captured during a match or practice and the like.

In this chapter, the focus will be on handball, a team indoor sport played with a ball by two teams with seven players, one of whom is the goalkeeper. To analyze handball sports videos, different CV tasks can be combined. For example, the object or person detection can be applied to detect the players on the field, the object tracking to follow the players' movements across the field, and action recognition to analysis of the players' performances.

In the next sections, a created dataset for handball will be presented, and then the simple CNN architecture and typical measures for evaluation of model performance are described. In the following sections, CV tasks will be described and implemented in the context of handball. Object detection with YOLO and Mask R-CNN is presented in Section 5, object tracking with Hungarian algorithm and Deep SORT in Section 6, and action recognition using LSTM model in Section 7. Applications of optical flow and spatiotemporal interest points for temporal segmentation and active player determination are presented in Section 8.

2. The dataset

The handball dataset used for the following experiments was recorded during a handball school, where participants were young handball players and their coaches.

The dataset consists of high-quality video recordings of practices and matches, filmed in a sport hall or in an outdoor handball field, without additional scene preparation or player instruction to preserve real-world conditions. The recordings were made using different stationary cameras positioned on the left or right border of the field on a tripod at 1.5 m, or from the spectator's viewpoint at the height of approximately 3.5 m and the distance from the filed limit of 10 m. The recordings are in full HD resolution (1920x1080) and contain from 30 to 60 frames per second.

The dataset is quite challenging with a cluttered background, a variable number of players at different distance from the camera, who move fast and often change direction, are often occluded with another player, have jerseys of similar color to the background, etc.

The data needs to be prepared, processed and labeled for each specific task that will be considered here, so that domain- and task-specific models can be made by either training from scratch if there is sufficient data, or preferably, by tuning an existing model for a similar task using examples from the new domain.

For the player and ball detection task, 394 training and 27 validating images were extracted from the videos in the handball dataset and manually labeled, to form the PBD-Handball dataset [6].

To obtain the ground-truth data for the player tracking task, a subset of videos from the handball dataset was first processed using the YOLOv3 object detector, then with the DeepSORT tracker to bootstrap the annotation process, and lastly manually corrected. The total duration of the annotated dataset, named PT-Handball, prepared for this task is 6 min and 18 s [7].

For the action recognition task, parts of the videos containing the chosen actions were extracted from the whole handball dataset to get a subset of 2,991 short videos that were then labeled with one of the 10 action classes, or the Background class where action is not happening. This dataset is referred to as PAR_Handball.

3. Convolutional neural networks

Typical models used today for image classification and object detections tasks are based on Convolutional Neural Networks (CNNs), since they are adapted to solve the problems of high-dimensional inputs and inputs that have many features. The CNN network consists of a number of convolution layers, after which the network has been named, the activation and pooling layers, and one or more fully connected layers at the end of the network [8], (**Figure 2**).

The convolution layer refers to a mathematical operator defined over two functions with real value arguments that give a modified version of one of the two original functions. The layer takes a map of the features (or, in the first layer, the input image) and convolves it with a set of learned parameters resulting in a new two-dimensional feature map. The sets of learned parameters (weights and thresholds) are called filters or kernels. Each filter is a 2D square matrix, small in size compared to the image to which it is applied (equal depths as well as the input). The filter consists of real values that represent the weights that need to be learned, so that the output feature map contains useful information such as a particular shape, color, edge in order to give the network good results.

The pooling layer is usually inserted between two successive convolution layers to reduce a map resolution and increase spatial invariance or network insensitivity to minor shifts such as rotations, and translations of features in the image. The pooling layer also reduces memory requirements for network implementation. The most commonly used pooling methods are arithmetic mean and maximum, but several other pooling methods are also used in CNN architecture, such as Mixed Pooling, Stochastic Pooling, Spatial Pyramid Pooling and others [8].

The activation function defines the output of a node given an input or set of inputs. In its simplest form, this function is binary and represents the action potential of neurons by propagating the output value of the neuron or by stopping it. There is a broad range of univariate functions of linear combination of the input variables acting as CNN activation functions such as linear activation functions, jump functions, and sigmoidal functions. The jump and sigmoidal functions are a better choice for neural networks that perform classification tasks while linear functions are often used in output layers where unlimited output is required. Newer architectures use activation functions, typically Rectified Linear Unit (ReLU), behind each layer.

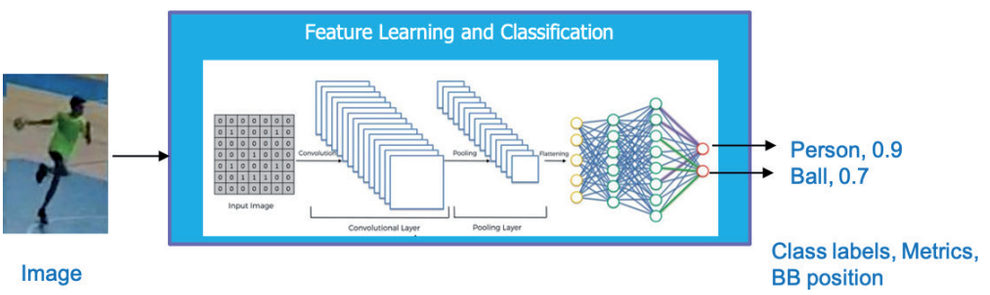

Figure 2.
Simplified CNN architecture.

A fully connected layer is the last layer in the network. The name comes because of its configuration: all neurons are linked to all the outputs of the neurons in the previous layer. Fully connected layers can be viewed as special types of convolution layers where all feature maps and all filters are 1 x 1.

Network hyperparameters are all parameters needed by the network that have to be set before the network is provided with data for learning. The hyper-parameters in convolutional neural networks are the learning rate, the number of epochs, the number and kind of network layers, the activation function, the initialization weights, input pre-processing and the error function.

Selecting the structure of the CNN network for feature extraction plays a vital role in object detection because the number of parameters and types of layers directly affect the memory requirements, speed, and performance of the detector.

In this paper, two types of CNN-based networks, YOLO and Mask-RCNN, have been used for object detection, while for the task of action recognition, the CNN network is not used on its own, but it forms a part of the more advanced LSTM network as the feature extractor.

4. Evaluation measures

The performance of object detectors is usually evaluated in terms of accuracy, recall, precision, and F1 score [9], for a given confidence threshold. The same measures can also be used for evaluation of the action classification task.

The detections are deemed true positive when the intersection over union of (IoU) the detected bounding box and the ground truth box is greater than 0.5. The IoU measure is defined as the ratio of the intersection of the detected bounding box and the ground truth (GT) bounding box and their union, see **Figure 3**.

Since the confidence threshold controls the tradeoff between recall and precision, Average Precision (AP) measure is frequently used to evaluate the performance of the detectors. The AP is the area below the precision-recall curve which is calculated for every class by varying the confidence threshold. To get the mean Average Precision (mAP) value, mean of AP values of all classes is calculated.

Since there is no single measure that can uniquely describe the complex behavior of trackers, several measures are used to evaluate the tracking performance. These measures are the number of identity switches (ID), identification precision (IDP), identification recall (IDR) and the identification F1 (IDF1) measures [10].

An identity switch occurs when an object that was assigned an ID j in previous frames, gets a new id k, $k \neq j$ in a subsequent frame. The IDF1 measure focuses on how long a target is correctly identified, regardless of the number of mismatches. It is the ratio of correctly identified detections over the average number of ground-truth and computed detections.

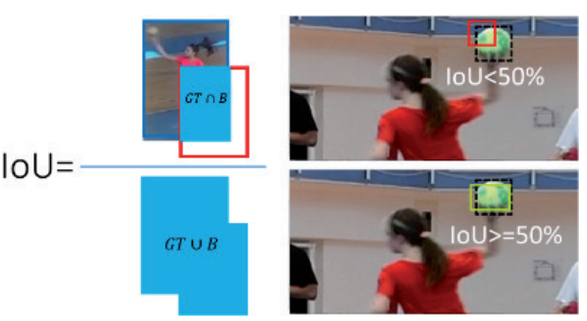

Figure 3.
Visualization of intersection over union (IoU) criteria equal to or greater than 50%.

5. Object detection

The task of object detection is to find instances of real-world objects in images or videos. A detected object is typically marked with a bounding box and labeled with a corresponding class label and classification confidence value. Thus, object detection includes both the problems of finding the location of the object on the scene and of classification for predicting the class to which the object belongs to.

In case of player detection, the object detector should be able to overcome challenging conditions such as variable number of players, different player positions, varying distance of the player from the camera, the possibility of changing shape and appearance of players in time, presence of the blur of due to the speed of the movement, occlusion, shadows of artificial and external light, as well as cluttered background.

Nowadays, the focus in object detection is on CNNs that have been extended to be able to both detect and localize individual objects on the scene. In the following subsections, two different object detectors YOLO and Mask R-CNN are described with a corresponding experiment of player and ball detection.

5.1 YOLO

YOLO is a detection algorithm based on a single-stage CNN architecture that can detect multiple objects in an image in real-time. The main idea is to predict bounding boxes and confidence values for grid cells into which an image or frame is divided. In the cases when an object is spread across more than one grid cell, the holder of its prediction will be the center cell.

There have been four versions of YOLO since it was first published. In the original version, the network architecture has 24 convolutional layers with two additional fully connected layers. The purpose of the convolutional layers is the feature extraction, while for fully connected layers to calculate the bounding boxes predictions and probabilities. The bounding box predictions and class probabilities are associated with grid cells so that if an object occupies more than one cell, the center cell will be designated to be the holder of prediction for a particular object [11].

In the next version, YOLOv2 [12], five convolution layers were replaced with max-pooling layers, and instead of the fully connected layers, predefined anchor boxes are introduced. In the training phase, to define the anchor boxes, YOLOv2 uses k-means clustering on ground-truth bounding boxes where boxes translations are relative to a grid cell.

YOLOv3 [13] is the third version of the YOLO object detector. It consists of 53 convolutional layers of (3 × 3) and (1 × 1) filters with shortcut connections between layers (residual blocks) used for feature extraction. The last convolutional layer predicts the bounding boxes, the confidence scores, and the prediction class. It predicts possible bounding boxes at three different scales using a structure that is similar to feature pyramid networks. In this way three sets of boxes are predicted at each feature map cell for each scale, to improve the detection of objects of different sizes.

5.1.1 Player detection with YOLO

Player and ball detection performance of the YOLOv3 detector was tested on the handball dataset using two different models. The reference model, further marked as Y, is the pre-trained YOLOv3 model with 608 x 608 input image size with weights pre-trained on the COCO dataset and no additional training. The pre-trained model contains the person and sports ball among other classes from the COCO dataset. Transfer learning [14] was used to avoid training the models from the beginning.

The second model (YBP) was trained using transfer learning, on PBD-Handball part of the dataset. The input image resolution was increased to 1024 x 1024 from 608 x 608 of the original model and the model was trained for approximately 80 epochs. **Figure 4** shows an example of detection results for the "person" class.

To evaluate the performance of a model, the average precision (AP) metric of both classes and the mean average precision are used and shown on **Table 1**.

The best results for ball detection in terms of AP were achieved with the YPB model, which was trained on additional examples for both ball and person class and had an increased input image size. A small amount of training data can significantly improve detection results as can be seen in the example of ball detection which improved for 23%. The achieved results are satisfactory given the demanding environment but are not sufficient for commercial application, so the training dataset should be increased.

5.2 Mask R-CNN

Mask R-CNN [15] is a two-stage CNN that can not only detect and localize multiple objects simultaneously present in the image, but also provides a segmentation mask of the objects, that is, assigns a membership value to each of the pixels belonging to the object. The first stage of the network is a region proposal network that finds the regions of the image that are likely to contain objects (regions of interest, RoI) and proposes candidate object bounding boxes. A sliding window is applied to the feature map to examine the probability whether there is an object class or a background in the examined region. Then, bounding boxes and masks are generated with the corresponding confidence values for all possible classes.

In the second stage, there are two parallel branches of the network, a fully convolutional branch for predicting the segmentation masks and a fully connected branch used on each RoI for classification and for adjusting the proposed box size.

There are similar networks like R-CNN, Fast R-CNN, Faster R-CNN [16] on which Mask R-CNN is based to look up for object detection purpose.

Figure 4.
Player detections in handball scene with YOLOv3 (bounding boxes with confidences).

	PBD-Handball		
Model	**Ball AP**	**Person AP**	**mAP**
Y	13.53	66.13	39.83
YPB	35.44	63.77	49.61

Table 1.
Evaluation of the object detector.

Object Detector/Measure	Inference time / frame	Recall	Precision	F1
YOLOv3	0.04 s	68%	95%	79%
Mask R-CNN	0.3 s	76%	98%	85%

Table 2.
Results of player detection with mask R-CNN and YOLOv3.

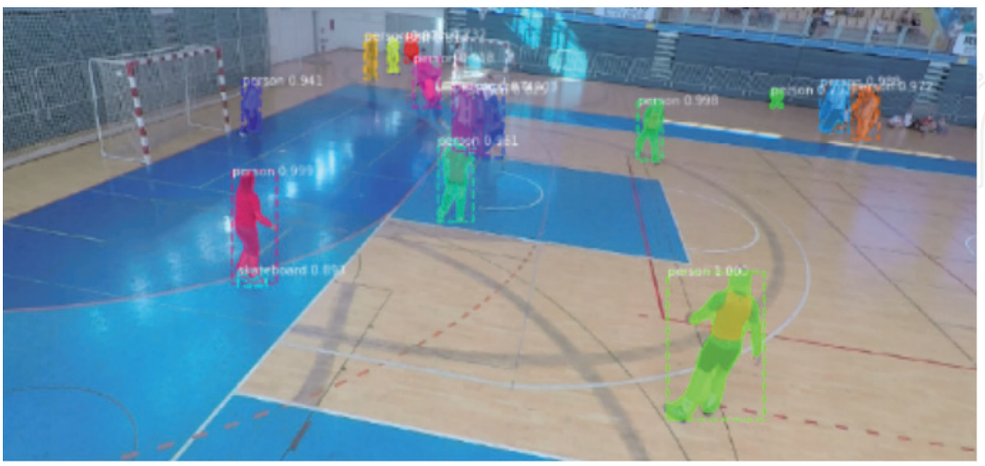

Figure 5.
Player detections obtained with mask R-CNN in a handball scene (bounding boxes with segmentation mask and confidence value).

5.2.1 Player detection with mask R-CNN

The performance of the Mask R-CNN for player detection was tested on the PBD-Handball dataset using the standard Resnet-101-FPN network configuration with pre-trained parameters on the COCO dataset. For player detection experiment, only the bounding boxes that refer to the "person" class were considered.

To obtain a good balance of high detection rates and low false positive detections, detections with confidence values below a threshold experimentally set to 0.55 were discarded. The detector performance was evaluated in terms of recall, precision, F1 scores and inference time per frame (using the NVIDIA 1080ti GPU). The results and comparison with the YOLOv3 detector are shown in **Table 2**. Detection was considered as true positive when the intersection of the detected bounding box and the ground truth box was above 50%.

One handball scene with the bounding boxes, class confidence value, and segmentation masks obtained with Mask R-CNN is shown on **Figure 5**.

It can be concluded that the results of both the YoloV3 and Mask R-CNN detector are good enough to be used for further analysis of player performance. However, the YOLOv3 detector is much faster, so it can be used not only for offline analysis of recordings, but also for real-time detection, at the cost of somewhat reduced recall. The detection results could be improved if more data is used, but the performance depends on the number and size of the players on the scene, the contrast between a player and a background, illumination, etc.

6. Object tracking

Tracking of handball players in video is an example of a Multi-Object Tracking (MOT) problem, where the goal is to track both the position and the identity of

multiple objects present in video, so that the same unique ID is assigned to each object in every frame it appears. In an ideal case, in every video frame, all the present players should be detected in their correct position and a unique ID for each player, that stays the same throughout the video, should be assigned. This is a difficult task as many players can be on the field, from 14 to 25, depending if it is a practice or a match, and every one of them needs to be tracked. Furthermore, players can leave and re-enter the camera field of view, move very quickly, often change directions, occlude each other and wear similar clothes, or clothes with similar color as the background [17].

Thanks to improvements in performance of object detectors, and thanks to the ability to deal with challenges such as cluttered scenes or dynamics of tracked objects, tracking-by-detection has become a leading paradigm for MOT.

When using tracking-by-detection, the tracking algorithm relies on the object detector to detect and locate the objects on the scene in each frame, while the role of the tracking algorithm itself is reduced to the problem of associating the detections across frames that belong to the same object. To do so, the tracker may use the information about bounding boxes obtained by object detection, such as their dimensions, the locations of their centroids, the relative position to the boxes in previous frames, or some visual features extracted from the image.

6.1 Hungarian assignment algorithm

The Hungarian algorithm [18] solves the problem of finding the globally optimal assignment of IDs to detected player bounding boxes, with respect to some cost function that is defined for an individual assignment. Here the cost function is defined only in terms of the parameters of the bounding boxes detected by the object detector in the current and previous frame, without using any visual features extracted from the video frames. Its value depends on the Euclidean distance of each detected object's bounding box centroid from the predicted centroid of an object in the track, and on the size difference of the bounding box and the last assigned bounding box to the same track.

Formally, the assignment cost $\mathrm{d}(b,k)$ of assigning a bounding box b with the centroid C_b and area P_b to the k-th track with the predicted centroid $C_{k+1}{}'$ is:

$$\mathrm{d}(b,k) = w \cdot d_2\left(C_b, C_{k+1}{}'\right) + (1-w)\left|P_b - P_{k-1}\right| \tag{1}$$

$$w \in [0,1], k \in N$$

where P_{k-1} is the area of the last bounding box assigned to the track with the ID k.

For the prediction of the centroid location $C_{k+1}{}'$, last known position of object centroid in the track k is used, so $C_{k+1}{}' = C_{k-1}$.

Moreover, a unique track ID is assigned to each detected bounding box whose detector confidence is higher than a set threshold during the initial assignment of bounding boxes to tracks. Afterwards, whenever the number of detected objects exceeds the number of currently active tracks, new tracks are created and initialized using the unassigned object's bounding box.

An existing track is considered inactive when no detections are assigned to that track for several frames. Once a track is marked inactive, no further detections are added to it, so if an object later reappears or is detected again, it will get a new track ID and will be considered a new object.

6.2 Deep SORT

Deep SORT [19] is a tracking algorithm that builds upon the Hungarian algorithm, adding the appearance information about the tracked objects into consideration when associating new detections with previously tracked objects. The appearance information is particularly useful for re-identifying players that were occluded or have temporarily left the scene. As in the previous case, a unique track ID is assigned to each bounding box within the first frame, and the Hungarian algorithm is used to assign the new detections to existing tracks so that the assignment cost function reaches the global minimum.

The cost function consists of the spatial distance $d^{(1)}$ in form of Mahalanobis distance of the detected bounding box from its position predicted according to its last known position, and a visual distance $d^{(2)}$ that compares the appearance of the detected object with the history of appearances of the tracked object. Formally, the cost function $c_{i,j}$ of assigning a detected object j to a track i is given by:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \tag{2}$$

where λ is a tunable parameter that determines the relative influence of the spatial distance $d^{(1)}$ and the visual distance $d^{(2)}$.

The spatial distance $d^{(1)}$ is given by the expression:

$$d^{(1)}(i,j) = \left(d_j - y_i\right)^T S_i^{-1} \left(d_j - y_i\right) \tag{3}$$

where y_i and S_i represent the mean and the covariance matrix of bounding box observations for the ith track, and d_j is the jth detected bounding box.

The visual distance $d^{(2)}$ is is given by the expression:

$$d^{(2)}(i,j) = \min\left\{1 - r_j^T r_k^{(i)} \,|\, r_k^{(i)} \in \mathcal{R}_i\right\}, \tag{4}$$

where r_j is the appearance descriptor extracted from within the jth detected bounding box, and $\mathcal{R}_i$ is the set of the last 100 appearance descriptors r_k^i associated with the ith track.

The $d^{(2)}$ measure uses the cosine distance between the jth detection and a number of detections already assigned to ith track, so if a visually similar detection was previously seen, the distance will be low.

The appearance descriptors are extracted using a wide residual neural network comprising two convolutional layers followed by six residual blocks that output a 128-element vector, and then normalized to fit within a unit hypersphere so that the cosine distance can be used. The network was pre-trained on a person re-identification dataset of more than a million images of 1261 pedestrians. The appearance information helps with re-identification of objects that have not been tracked for some time because of missed detections, because they were under occlusion or because they have briefly left the scene.

New tracks are formed whenever there are more detections in a frame than there are existing tracks or when a detection cannot be assigned to any track, because its spatial or visual distance is too far from any existing track. The maximum allowed $d^{(1)}$ and $d^{(2)}$ distance when an assignment is still possible is set with tunable thresholds.

6.3 The player tracking experiment

The tracking of players previously detected with the YOLOv3 detector using pre-trained tiny-yolo weights and confidence threshold set to 0.5 was performed on the PT-Handball dataset with the Hungarian algorithm and Deep SORT [20].

An example of a tracking situation in sequential frames when occlusions occurred is shown in **Figure 6**. The numbers above the bounding boxes represent the tracking ID of each player. The shown situation is quite demanding, resulting in rather unstable and inconsistent tracking in the selected frame. The Hungarian algorithm successfully tracked one of four players, and two of the players got new IDs after occlusion, while one player has switched ID with another, so that 807 got previously existing ID 812 (**Figure 6**, top row). Deep SORT managed to track correctly all four players (**Figure 6**, bottom row).

Since the best results were obtained with the Deep SORT algorithm, its ability to assign the correct IDs to detections is analyzed in more detail using the common MOT evaluation measures. The results are shown in **Table 3**.

For each player that should be tracked, the identity switches caused, 5 to 6 additional tracks on average, so there are 5 times more tracks than in ground-truth annotated data. Furthermore, a large number, precisely 1483, of identity switches are present, due to a relatively large number of players in the video that move fast, exit the camera field view, frequently change positions, and occlude each other.

The number of players that are simultaneously present in the frame obviously affects the tracking performance, and according to the IDF1 measure, the players can be correctly identified for 24,7% of the time.

Tracking mistakes can be attributed to several factors. As in all tracking-by-detection algorithms, the accuracy of tracking is greatly influenced by the accuracy of the object detector. If a player is inaccurately detected, the tracking will be inaccurate as well. Furthermore, the scale of an object, occlusion, and the similar color of the players' clothes with the background often cause tracking problems. To overcome these problems, in further work, multiple camera systems will be investigated [21], which can also allow for a more robust generation of a top-view trajectory [22].

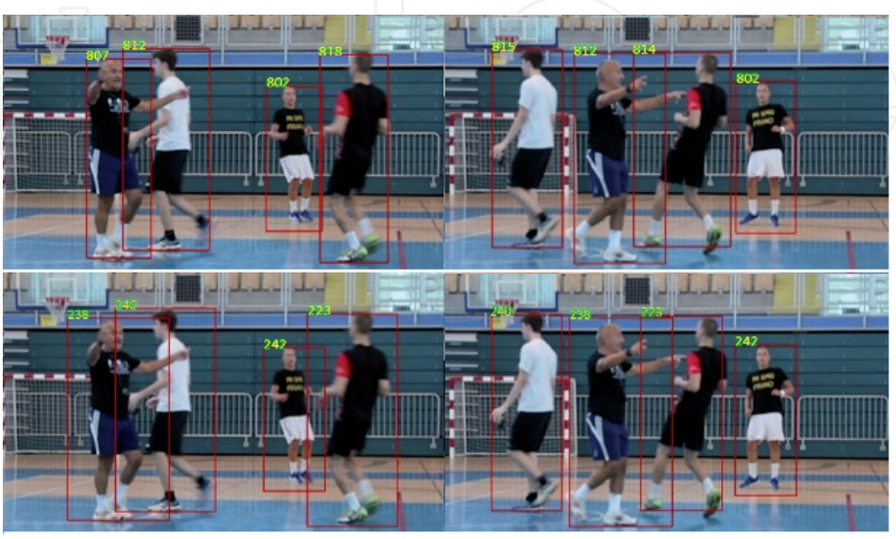

Figure 6.
An example of tracking situation with occlusion. Top: Hungarian algorithm, bottom: Deep SORT. The left and right frames are 1 second apart.

Measure	Value
#tracks in the ground truth	279
#tracks	1554
Identity switches	1483
IDF1	24.7%

Table 3.
Performance evaluation of deep SORT.

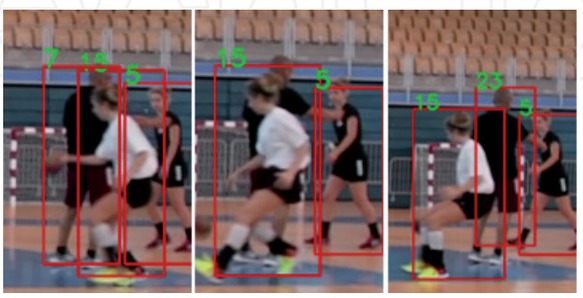

Figure 7.
Problem of re-identification after occlusion.

In **Figure 7**, the top row shows the problem of re-identification after occlusion, and the bottom row the problem of identity switch due to small scale and similar colors.

7. Action recognition

The goal of action recognition is to infer which action takes place in a set of image or video observations. Some simple actions, like eating or cutting, could be recognized using just a single frame, but actions are mostly much more complex and take place over a period of time, so they need to be analyzed across consecutive video frames.

Here, for recognition of handball actions, a simple long short-term memory (LSTM)-based artificial recurrent neural network is used.

During the handball game, every player is moving around the field performing different actions with a ball, such as shot, jump-shot or dribbling, or without the ball, such as running or defending. Some actions are performed by more players such as passing the ball or crossing.

7.1 LSTM

Unlike CNNs and other so-called feedforward neural (FFNN) networks, recurrent neural networks (RNNs) [23] have connections that feed the activations of an input in a previous time step back into the network, to influence the output for the current input. These activations from the previous time step are held potentially indefinitely in the internal state of the network, so the temporal context is not limited to a fixed window that could be used as an input to a FFNN. This property makes RNNs especially appropriate for modeling sequences, such as text or a sequence of video frames in action recognition.

Long Short-Term Memory (LSTM) [24] is a type of recurrent neural network (RNN) designed to model temporal sequences and their long-range dependencies

more accurately than conventional RNNs. In the recurrent hidden layers, the LSTM contains special units called memory blocks. Those units contain memory cells, that have self-connections storing the temporal state of the network, and gates, which are special multiplicative units that control the flow of information. The flow of input activations into the memory cell is being controlled by the input gate, while the output gate controls the output flow of cell activations into the rest of the network. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, thus causing the adaptive forgetting or resetting the cell's memory.

7.2 The action recognition experiment

In the experiment handball actions from the PAR-Handball dataset are considered: Throw, Catch, Shot, Jump-shot, Running, Dribbling, Defense, Passing, Double-pass, and Crossing. An example of jump-shot action is shown in **Figure 8**. The action consists of a sequence of different phases of jump-shot action from running, take-off, flight, throw, and landing that are captured on different video frames.

Different actions can take different amounts of time to perform, so the average number of frames in a video depends on the action class it belongs to, as shown in **Figure 9**.

Only Throw and Catch actions are significantly shorter (two or three times) than the other action classes that have a duration of around 60 frames on average.

Because these two actions are also parts of the more complex ones, like Passing, the model is trained once with all 11 classes and once with 9 classes, excluding Throw and Catch.

The model selected for action recognition is a LSTM-based network with one LSTM layer with 1,024 units, followed by a dropout layer with 0.5 dropout, one

Figure 8.
Active player collage for jump-shoot action.

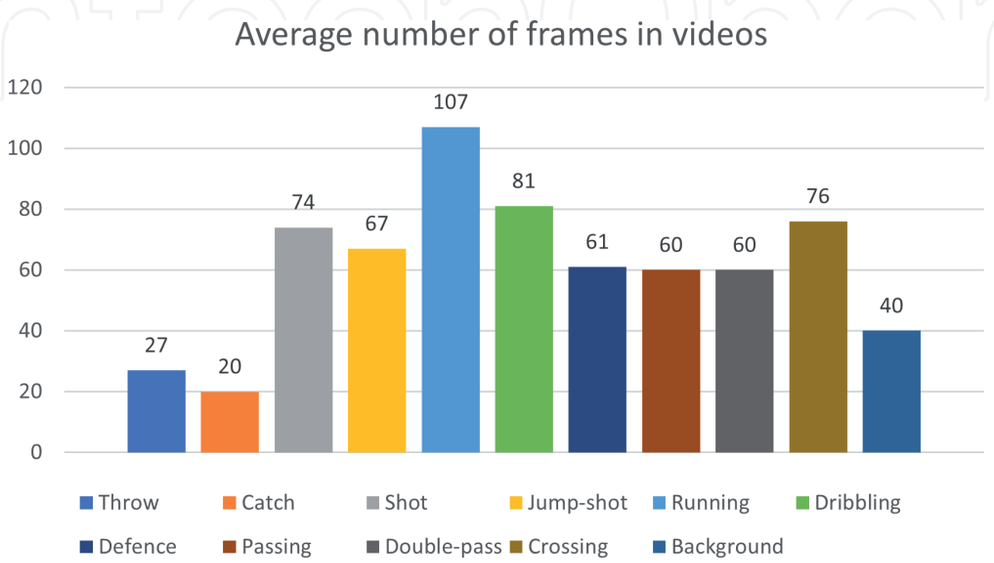

Figure 9.
Average number of frames per action from the handball dataset.

fully connected layer with 512 neurons, also followed by a dropout layer with 0.5 dropout rate. and the output layer with 11 neurons.

The input to the LSTM consists of a sequence of features extracted from video frames using the InceptionV3 [25] network with the ImageNet [26] re-trained weights as the starting point. The model is trained with the Adam optimize with a learning rate of 0.00001 and decay of 10–6 for up to 100 epochs, stopping early if the validation loss does not improve for more than 20 epochs.

Different frame selection strategies and different input size of sequences from 20 to 80 were used to train the model, because actions might have most distinctive characteristics in different parts of the sequence.

In videos containing more frames than expected, the chosen number of frames were selected consecutively from either beginning, middle or the end of the video, or from the whole video by decimation, i.e., by skipping some frames at regular intervals. Conversely, copies of existing frames were inserted between frames to extend the number of frames.

The action recognition results obtained by the described LSTM model, considering the frame selection strategies and different class numbers are shown in **Figure 10** in terms of validation accuracy.

Having to classify a smaller number of classes can generally be considered a simpler task, so, as expected, the models trained on 9 classes have better results on average than the models trained on 11 classes. However, the best result overall of 70.94% is obtained for the model with 11 classes and 45 frames taken from the middle of the sequences. This is possibly due to the overlap of some actions that make them more difficult to recognize, as the actions Throw, and Catch that are parts of other actions such as Passing, Double-pass, Crossing, Shot and Jump-shot. Closely behind with 70.55% is the model trained on 9 classes with 20 frames in the middle, and in the third place is the model trained on 9 classes with last 45 frames with 70.47% validation accuracy.

Taking into consideration only the number of input frames and ignoring the number of classes or frame selection, the best results are obtained with 45 frames followed by 20 frames. In most cases the additional frames in the sequence do not

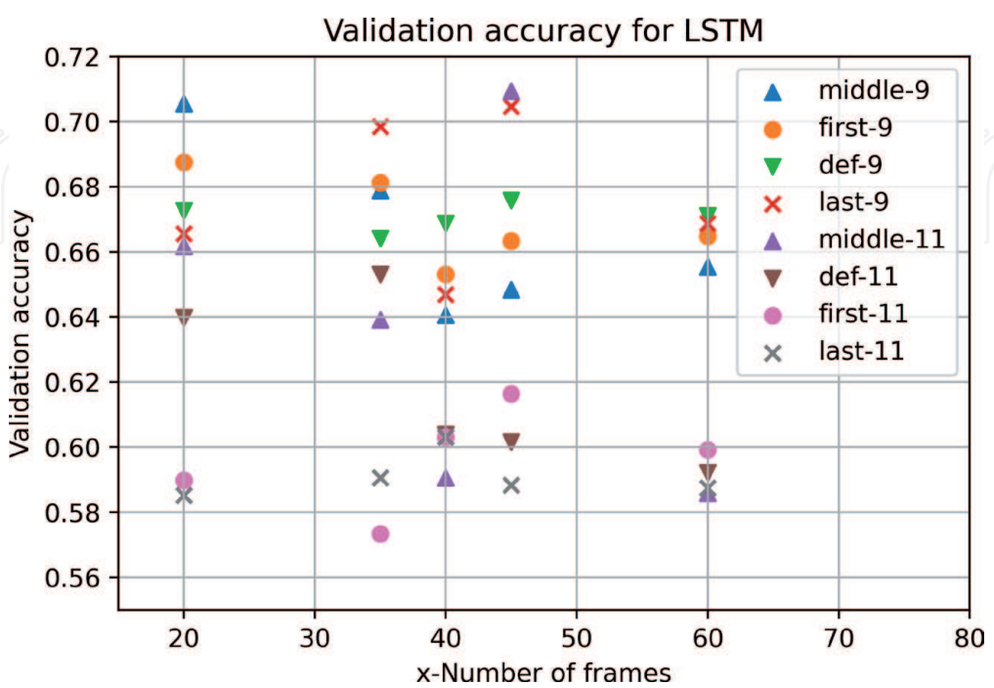

Figure 10.
Validation accuracy for different lengths of input sequences and 9 or 11 action classes.

improve the result much over the models trained with 20 frames. Considering only the way the sequence is selected, the highest average accuracy of 67.69% is achieved by the model while trained on 9 classes and the last frames selection, followed by 67,05% by skipping frames.

It can be noted that regardless the strategy of selecting frames, increasing the number of input frames does not contribute to a better result. The number of frames and frame selection strategies appear to be highly dependent on the type of action being performed.

8. Application of low-level video features

Low-level features extracted from video frames, combined with specific knowledge about the problem domain can sometimes be used for solving specific tasks and generate conclusions about the objects in the image or for scene analyzes. For example, optical flow can be used as a measure of motion in video, and for rough temporal segmentation of the input video in order to automatically cut periods of inactivity or detect intervals of repetition of a certain exercise in handball training.

If the low-level features such as optical flow or spatio-temporal interest points are used with additional information such as the detected player bounding boxes, conclusions can be drowned about the most active player on the scene and automatic detection of players that are at a certain time likely to be performing the action that is of most interest for the interpreting the scene [27].

8.1 Temporal segmentation using optical flow

A low-level feature that captures motion information is optical flow, which is estimated from the time-varying image intensity. A moving point on the image plane produces a 2D path $\boldsymbol{x}(t) \equiv \left(x(t), y(t)\right)^T$ in camera-centered coordinates, and the current direction of movement is described by the velocity vector $d\boldsymbol{x}(t)/dt$. The 2D velocities of all visible surface points form the 2D motion field.

The movement of points can be estimated from consecutive video frames, using some optical flow estimation algorithm, e.g., the Lucas-Kanade method [28]. This method assumes that small sections, i.e., groups of pixels of the initial images move with the same velocity, so the result is a vector field V of velocities of each image section. At each point $V_{x,y}$ in the field, the vector magnitude corresponds to the speed of movement and the vector angle represents the movement's direction.

A visualization of an optical flow field calculated between two video frames from the dataset is shown in **Figure 11**. The direction and magnitude of optical flow at each point is represented by the direction and length of each arrow.

In an uninterrupted recording of a handball training, there are usually periods of repletion of a certain exercise, where all players repeat the exercise either simultaneously or taking turns, followed by short pauses where the coach explains the next

Figure 11.
Two consecutive frames in video and the corresponding optical flow field.

exercise to be performed. The periods of higher activity, when players perform the exercises are characterized with the higher magnitude of extracted motion features from video, while the periods when players queue or wait for instruction (**Figure 12**) are characterized with lower magnitude of motion features [29].

To mark the periods of inactivity and segment the videos into sections where a single exercise is repeatedly practiced, an optical flow threshold is used. First, the optical flow field is calculated between two consecutive frames sampled each N frames (here, N = 50). Then, mean optical flow magnitude is calculated for each field, resulting in a single value for each sampled time point in video. The video is cut at time points when the mean magnitude of optical flow is lower than an experimentally determined threshold value. An example of the mean optical flow magnitude calculated for a video sequence where there were short pauses of 10–20 seconds between active repetition of an exercise was is shown in **Figure 13**. It can be seen that the normalized flow threshold of about 0.07 clearly separates the periods of inactivity from parts of video showing exercise.

8.2 Active player determination

In a typical footage of a handball game or training session, at a given time only one player or a small proportion of players present on the scene participate in the action that is currently in focus, e.g., jump-shot, passing, while others may perform actions that are not currently relevant for interpreting the situation, e.g., moving into their positions. To train the action recognition models, the actions of those players that perform the action of interest should be annotated. The annotation process is at least partly manual, so it is time-consuming and tedious given the large amounts of video data to process. To assist with annotation, a method is proposed to select among the detected and tracked players, the ones that are currently the

Figure 12.
A typical training situation. Two players on the right are performing the current task, while the rest are queuing.

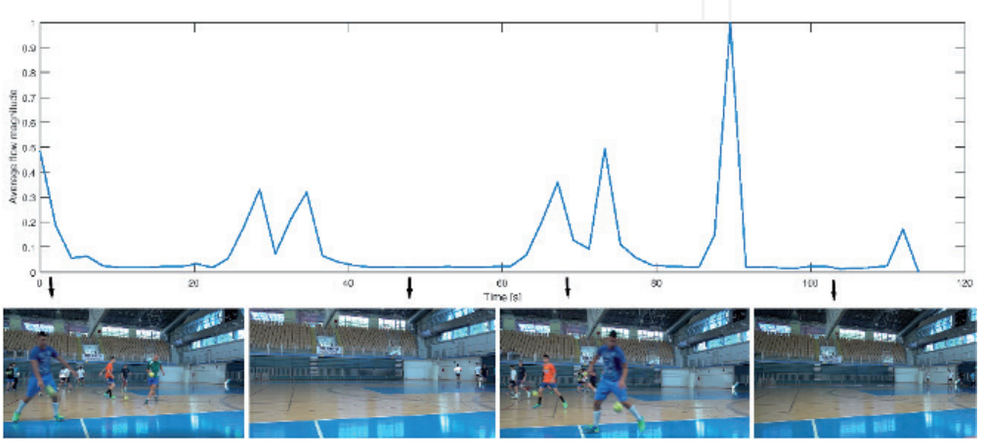

Figure 13.
An example of segmentation of a video sequence using optical flow magnitude.

most likely to be performing the action of interest, here called active players. Thus, instead of reviewing every single players' activity at all times, the manual annotation is reduced to verification of only the proposed active players' tracks.

First, the players are detected and tracked as described in previous chapters. Then, the information about player positions, i.e., the detected bounding boxes, is combined with the low-level movement features, such as optical flow or spatiotemporal interest points, to obtain a measure of each player's activity in the considered time interval.

The optical flow-based measure of player's activity (A_b^{OF}) is calculated using the bounding box (B_b) of each detected player bounding box, as the maximum optical flow magnitude within that box:

$$A_b^{OF} = \max_{B_b} \left| V_{x,y} \right|; x, y \text{ within } B_b \tag{5}$$

An alternative feature to optical flow for defining the activity measure are spatiotemporal interest points, or STIPs. STIPs are an extension from the spatial domain into both spatial and temporal domain of the notion of interest points in images, which are points with a significant local variation of image intensities. STIPs are thus points in the image with large variation of values in both spatial and temporal directions around these points.

As for calculating the optical flow, there are several algorithms that can be used to detect the STIPs, e.g. the method presented in [30] is based on the Harris corner operator (Harris3D) that is extended into the spatiotemporal domain, [31] uses a Gaussian filter in the spatial domain and a Gabor band-pass filter in the temporal domain, the algorithm presented in [32] is based on Hessian3D derived from SURF, and the selective STIPs detector [33] is designed to specifically detect the STIPs that likely belong to persons and not to the background.

Given that movement during the performance of various sports actions causes significant variation of appearance in that part of image, it is expected that there will be more detected STIPs in image regions with more intense player's activity [34].

An activity measure based on density of STIPs in the area near the detected player, A_b^{STIP}, can be calculated for a player with a bounding box B_b and area P_b as:

$$A_b^{STIP} = \frac{\#STIP}{P_b}; STIP \text{ within } B_b. \tag{6}$$

In the experiment here, the Harris3D detector with the default parameters was used to extract the STIPs. **Figure 14** shows an example of the detected player bounding boxes and STIPs in a video frame.

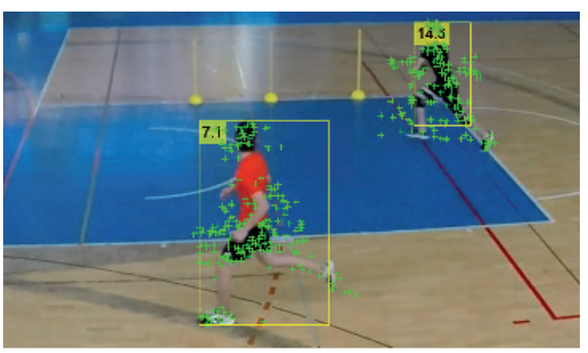

Figure 14.
Detected players and spatiotemporal interest points.

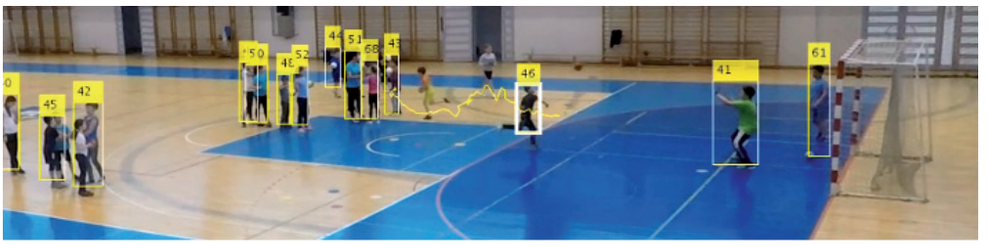

Figure 15.
Detected leading player (white box) and his trajectory through the whole sequence (yellow line).

A threshold of activity measure can be used to filter active from inactive players, since the players that perform sports actions should make more sudden movements corresponding to higher activity measures than other players.

Looking at activity measures in a sequence of frames, the ranking of players' activity can change between frames. So, to choose the players that are active throughout the sequence, the Active player score is calculated as the average activity measure of the player along the trajectory of the player's bounding boxes. The result is a set of player trajectories with corresponding player activity scores (**Figure 15**).

9. Conclusion

In this chapter, the applications of deep learning methods on typical CV tasks such as object detection, object tracking and action recognition are presented on videos from the handball domain, recorded during training and matches.

Handball is a team sport, played with a ball, with well-defined player's roles, goals and rules. During the game, the athletes move quickly throughout the field, change positions and roles from defensive to offensive and vice versa, use different techniques and actions, and doing so often get partially or completely occluded by another athlete, making player detection, tracking and action recognition challenging problems of ongoing research interest.

For detection, the algorithm must be able to locate an object in relation to its environment and, define that object. It is important for the detector to be as accurate and fast as possible especially if the real time detection is needed. State of the art deep learning-based detectors such as YOLOv3 and Mask R-CNN, prove to be successful for player detection, while the performance on ball detection still lags due to the combination of its small size, great speed and occlusion by the players.

Once objects such as players are detected, they can be tracked. Here, the Hungarian assignment algorithm and SORT with a deep association metric (Deep SORT) are considered for tracking. The goal of a tracker is to assign the same unique track ID to the same player in consecutive frames, which is complicated by the changes of appearance and sudden motions of players. Thus, the trackers can model this motion or the changing appearance to help the association process. The Deep SORT adds an appearance model based on deep neural network features. This appearance model allows the Deep SORT method to re-identify players that have been temporarily occluded or left the scene much more successfully than the other tested methods, making it more appropriate for use in the handball domain.

For the action recognition task, LSTM network is used, as it is suited to deal with both image information contained in a single video frame and its temporal evolution during the performance of actions. The obtained action recognition results are promising, however due to dependence of the action recognition model on the performance of previous stages, i.e. object detection and tracking, the challenge remains to improve all three tasks. As in all deep learning tasks, an important factor

is gathering enough training data, which can be facilitated by methods that reduce the manual effort of labeling ground truth data. To that end, the experiments for automatic temporal segmentation of the raw footage and a method for detecting the active player in a sequence using low level visual features were presented.

Advances in deep learning methods promise continued improvement in the analysis of dynamic sports scenes, in order to recognize more complex activities, plan competitive tactics and monitor player progress.

Acknowledgements

This research was fully supported by the Croatian Science Foundation under the project IP-2016-2106-8345 "Automatic recognition of actions and activities in multimedia content from the sports domain" (RAASS) and by the University of Rijeka under the project number uniri-drustv-18-222.

Author details

Marina Ivasic-Kos*, Kristina Host and Miran Pobar
Department of Informatics, Center for Artificial Intelligence and Cybersecurity, University of Rijeka, Rijeka, Croatia

*Address all correspondence to: marinai@uniri.hr

References

[1] S. J. D. Prince, Computer Vision: Models, Learning, and Inference, Cambridge Univesity Press, 2012.

[2] M. Nixon and A. Aguado, Feature Extraction and Image Processing for Computer Vision, Elsevier, 2020.

[3] K. Soomro and A. Zamir, "Action Recognition in Realistic Sports Videos," *Springer,* 2014.

[4] P. Pareek and A. Thakkar, *A survey on video-based Human Action Recognition: recent updates, datasets, challenges, and applications,* Springer Nature, Artificial Intelligence Review, 2020.

[5] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," in *International Conference on Engineering and Technology (ICET)*, Antalya, 2017.

[6] M. Buric, M. Pobar and M. Ivasic-Kos, "Adapting YOLO Network for Ball and Player Detection," in *8th International Conference on Pattern Recognition Applications and Methods*, 2019.

[7] K. Host, M. Ivasic-Kos and M. Pobar, "Tracking Handball Players with the DeepSORT Algorithm," in *9th International Conference on Pattern Recognition Applications and Methods*, Valletta, 2019.

[8] J. Johnson and A. Karpathy, "Convolutional Neural Networks," Stanford Computer Science, [Online]. Available: https://cs231n.github.io/convolutional-networks. [Accessed 11 3 2019].

[9] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *International Journal of Machine Learning Technology,* pp. 37-63, 2011.

[10] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara and C. Tomasi, "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking," 2016. [Online]. Available: http://arxiv.org/1609.01775.

[11] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[12] A. Farhadi and J. Redmon, "YOLO9000: Better, Faster, Stronger.," in *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, 2017.

[13] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr 2018. [Online]. Available: https://arxiv.org/1804.02767.

[14] D. Cook, K. D. Feuz and N. C. Krishnan, "Transfer learning for activity recognition: a survey," *Springer London,* 2013.

[15] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017. [Online]. Available: https://arxiv.org/1703.06870.

[16] S. Ren, K.He, R. Girshick and J.Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015. [Online]. Available: https://arxiv.org/1506.01497.

[17] P. L. Mazzeo, P. Spagnolo, *M. Leo* and T. D'Orazio, "Visual Players Detection and Tracking in Soccer Matches," in *IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, Santa Fe, 2008.

[18] H. W. Kuhn, "The Hungarian method for the assignment problem," in *Naval Research Logistics Quarterly*, vol. 2, 1955, pp. 83-97.

[19] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *International Conference on Image Processing (ICIP)*, Septe, 2017.

[20] M. Buric, M. Ivasic-Kos and M. Pobar, "Player Tracking in Sports Videos," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2019.

[21] T. D'Orazio, *M. Leo*, P. Spagnolo, P. Mazzeo, N. Mosca, M. Nitti and A. Distante, "An Investigation Into the Feasibility of Real-Time Soccer Offside Detection From a Multiple Camera System," *IEEE Transactions on Circuits and Systems for Video Technology,* pp. 1804-1818, Dec. 2009.

[22] G. Kayumbi, P. Mazzeo, P. Spagnolo, M. Taj and A. Cavallaro, "Distributed visual sensing for virtual top-view trajectory generation in football videos," in *7th ACM International Conference on Image and Video Retrieval*, Niagara Falls, 2008.

[23] L. Medsker and L. Jain, Reccurent Neural Networks, CRC Press, 2001.

[24] H. Sak, A. Senior and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," Google, USA.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *EEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[26] J. Deng, W. Dong, R. Socher, K. L. L. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, 2009.

[27] M. Ivasic-Kos and M. Pobar, "Building a labeled dataset for recognition of handball actions using mask R-CNN and STIPS," in *7th IEEE European Workshop on Visual Information Processing (EUVIP)*, Tampere, 2018.

[28] J. Barron, D. Fleet and S. Beauchemin, "Performance of optical flow technique," p. 43-77, 1994.

[29] M. Ivasic-Kos, M. Pobar and J. Gonzàlez, "Active Player Detection in Handball Videos Using Optical Flow and STIPs Based Measures," in *13th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2019.

[30] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis. 64,* p. 107-123, 2005.

[31] P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[32] A. Klaser, M. Marszalek and Schmi, "Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 2008-19th British Machine Vision Conference, British Machine Vision Association,* , Leeds, 2008.

[33] B. Chakraborty, M. Holte, T. Moeslund and J. Gonzàlez, "Selective spatio-temporal interest points," *Computer Vision and Image Understanding 116,* p. 396-410, 2012.

[34] M. Pobar and M. Ivasic-Kos, "Detection of the leading player in handball scenes using Mask R-CNN and STIPs," in *Eleventh International Conference on Machine Vision*, 2018.