ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MASTER DEGREE IN ARTIFICIAL INTELLIGENCE

MASTER THESIS
in
Machine Learning for Computer Vision

DEEP LEARNING MODELS FOR THE AUTOMATIC ANALYSIS OF VOLLEYBALL PLAYS

CANDIDATE
Giorgio Tsiotas

SUPERVISOR
Prof. Samuele Salti

Academic year 2020-2021
Session 3rd

Contents

Chapter 1

Introduction

1.1 Introduction

Collecting and analysing data is an important element in any field of human activity and research: through knowledge of the past, it's possible to predict the future and understand better the reality that surrounds us. In recent years, the development of technologies, the growth of computing power, the ability to collect and store huge amounts of data, has allowed the development and increasingly massive use of software that exploits complex mathematical models often ending up under the name of Artificial Intelligence models.

Even in sport, collecting and analyzing statistical data is Even in sport, collecting and analyzing statistical data is attracting growing interest. Some exemplar use cases are: improvement of technical/tactical aspect for team coaches, definition of game strategies based on the opposite team play or evaluation of the performance of players.

Other advantages are related to taking more precise and impartial judgment in referee decisions: a wrong decision can change the outcomes of important matches. Finally, it can useful provide better representations and graphic effects for example that make the game more engaging for the audience during the match.

In all high-level competition sports data is collected, for example, by counting the points scored or the type of action performed by each athlete. In the past when there were no alternatives, the collection of this type of data was done exclusively manually, thanks to the physical presence of a scout-man on the site of the sporting event: he transcribed everything that happened filling out a match report.

Nowadays it is possible to delegate this type of task to automatic software systems that can use cameras or even hardware sensors to collect images or

data and process them. Compared to single scout-men, these systems are more precise, faster and also collect much more data providing much more complex statistics in real-time.

One of the most efficient methods to collect data is to process the video images of the sporting event through mixed techniques concerning machine learning applied to computer vision.

As in other domains, in which computer vision can be applied, in sport, the main tasks are related to object detection, player tracking, and to the pose estimation of athletes.

Nowadays this kind of task is by mathematical models that are a subcategory of Artificial Neural Networks, called Convolutional Neural Networks (CNN). Thanks to their internal architecture CNNs are very good to filter important features inside the image to recognize specific shapes or objects.

Some of these models are already trained to perform this kind of task and are available online. Those can be downloaded and reused, re-adapting them with fine-tuning techniques to better cover the field of application in a specific topic.

The goal of the present thesis is to apply different models of CNNs on images of actions in volleyball matches, which is a team sport comparable to many others where a group of people shares a game court and plays with a ball.

Starting from video images of a volleyball match, we wanted to reproduce a bird's eye view of the playing court where are projected all the players are projected, reporting also for each player the type of action she/he is performing.

To train such a model, we used an online public database and expanded the labels on it.

We extracted the players positions and poses from each frames in numeric format. In the end, since we collected all the coordinates values of the player body joints in each frame of the sequences, we had a new dataset that lets us analyze the gesture data along the time and try to perform a gesture recognition model that we could use to label the action type, close to each player in the bird's eye view.

Usually, to have all this data, it is common to merge different cameras positioned in different points of view to rebuild the exact positions of each detected object.

The present work focuses on the recognition of these actions from video images of a single camera, which is less precise but can be retro-fitted to footage already acquired with standard equipment, e.g. TV recordings.

Chapter 2

Computer vision in Sport

2.1 State of the art

In sports, artificial intelligence was virtually unknown less than some years ago, but today deep learning and computer vision are making their way into a number of sports industry applications. Whether it is used by broadcasters to enhance spectator experience of a sport or by clubs themselves to become more competitive and achieve success, the reality is that the industry has substantially increased its adoption of these modern techniques [4]. Computer vision is interesting way to augment data collection because wearable tracking equipment and sensors are not an option in a lot of sport.

For team sport in which there is a court of game images are acquired through one or more cameras installed at close proximity of where the event takes place.

Usually the task for such sport is the ball and player tracking. To do that the calibration of the camera is the challenge. For fixed-angle cameras, this could be done through scene calibration, but broadcast cameras present additional challenges in that they often change their pan, tilt and zoom.

Computer vision has partially solved these limitations: thanks to machine learning, some systems are now able to distinguish between the ground, players and other foreground objects.

Methods such as colour-based elimination of the ground in courts with uniformly coloured surfaces allow computer vision models to detect the zones of a pitch, track moving players and identify the ball. For instance, colour-based segmentation algorithms are currently being used to detect the grass by its green colour and treat it as the background of the image or video frame, where players and objects move in front of it [4].

One of the key for computer vision in sports is player tracking. The

most advanced applications of computer vision in sport use automated segmentation techniques to identify regions that likely to correspond to players [4]. The results obtained from a computer vision system can be augmented by applying machine learning and data mining techniques to the raw player tracking data. Once key elements in an image or video frame are detected, semantic information can be generated in order to create context on what actions the players are performing (i.e. ball possession, pass, run, defend and so on).

These techniques can label semantic events, and be used for advanced statistical analysis of player and team performance. Suggestions can also be constructed on the optimal positions of players on the pitch and be displayed to coaches in a manner in which they can compare ideal player positioning against their actual positions in a given play.

The vast opportunities created from this player tracking technology has the potential to revolutionise training and scouting for players in sports [4].

In the *Modeling offensive player movement in professional basketball (Steven Wu, Luke Bornn) - 2017*, the authors say, referred to the introduction in 2013 of a A.I. computer vision system:

> *"player tracking data in all NBA arenas introduced an overwhelming amount of on-court information - information which the league is still learning how to maximize for insights into player performance and basketball strategy" [22]*

2.1.1 Tennis

In racket and bat-and-ball sports, such as Tennis, Badminton or Cricket, computer vision has been widely used since the mid-2000s. Ball tracking systems attempt to look through each camera image available to identify all possible objects resembling the characteristics of a ball.

An example of the use of computer vision in tennis can be spotted in one of the major tournaments in the sport. In 2017, Wimbledon partnered with IBM to include automated video highlights picking up key moments in the match by simply gathering data from players and fans, such as crowd noise, player movements and match data [4].

Tracking system is based on the principles of triangulation using visual images and timing data captured from high-speed cameras setup around the stadium; and the cameras are calibrated and synchronised prior to each event. The cameras are usually placed high above the courts in such a way that they can capture the trajectory of the balls with minimal obstructions. Although there has been some disputes with its line calling accuracy (up to

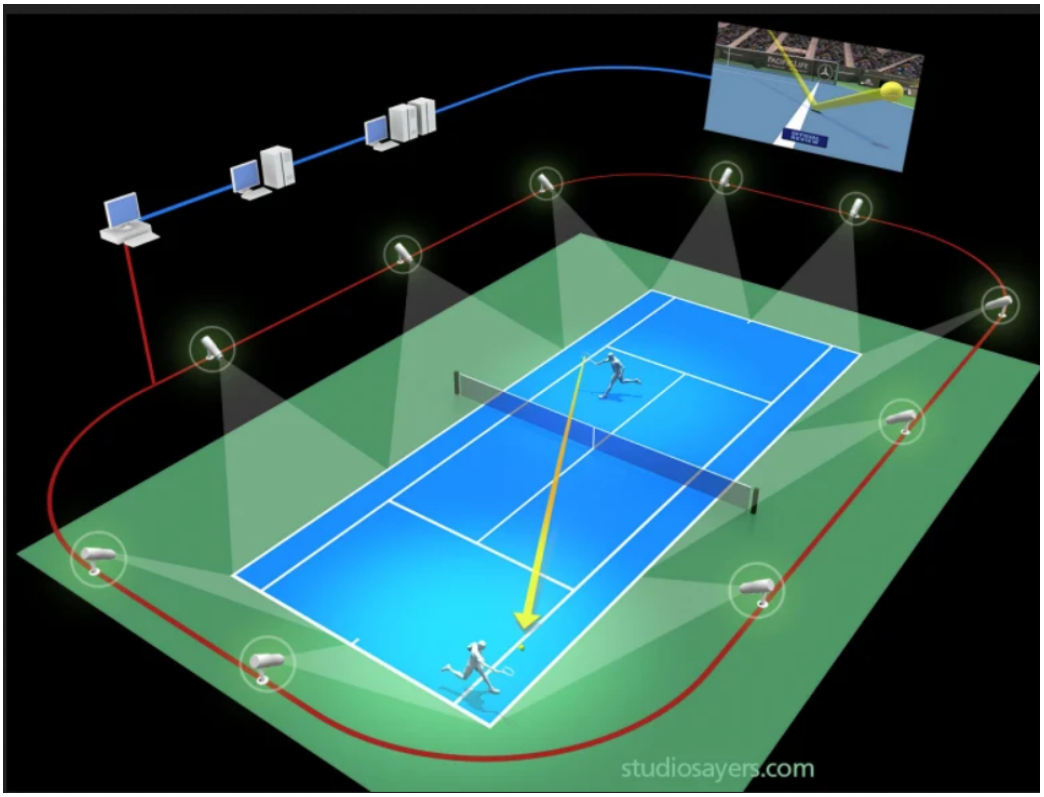

Figure 2.1: Representation of how cameras are positioned in a tennis court

3.6 mm error), it has been generally accurate except for some edge cases [3, 11].

In Tennis there were several portable applications working also as app on smartphones, maybe the conditions that there are 2 or at most 4 players in the court help to have a free visual field [3].

2.1.2 Auto racing

Argo AI/Ford Motor Company has used deep learning to develop self-driving cars and is now expanding its application of deep learning to help improve safety measures in the world of auto racing . They reportedly more accurate results than humans in ability to identify a car that is experiencing a malfunction during a race, alerting the driver before something bad happens [20].

2.1.3 Baseball

AI can be used for automation of sports journalism: the Associated Press is working with Automated Insights, a North Carolina-based startup to expand the media outlet's coverage of games in Minor League Baseball (MiLB).

Their platform translates hard data from MiLB into narratives, using natural language. As a result, AP has increased its reporting capacity to cover 13 leagues and 142 MLB-affiliated teams.

In general sports works well for automated journalism since sports stats are numbers-based. These data can be structured in a way which makes automated articles easy to write [20].

2.1.4 Football

In football, FIFA certified goal line technology installations in major stadiums using a 7-camera computer vision system. It uses a goal detection systems with multiple view high-speed cameras covering each goal area that detect moving objects by sorting potential objects resembling the playing ball based on area, colour and shape. With an accuracy error rate of 1.5cm and a detection speed of 1s, it enables football referees to immediately decide whether or not a ball has crossed the goal line and a goal should be awarded [4].

Recently the video technology support for referees in making decision is intensifying for all sports: also if there isn't any relations with a A.I. system, now we are used to the VAR (Video Assistant Referee) during the soccer matches.

In a lot of sport referees take decision thanks to the computer vision support

2.1.5 Basketball

Nowadays, statistics are starting to heavily influence strategy and player evaluation. Player evaluation has also changed with more complex statistics, that give a more accurate summary of the performance of the players. Optical tracking allows a huge amount of new, finely detailed data to be collected, causing an explosion of data and new analytic possibilities.

Every team in the NBA has the SportVu system implemented in their facilities to collect information from every game [21].

SportVu uses six cameras and is capable of tracking the (x,y) positions of the players along with the (x,y,z) position of the ball 25 times every second. This data is combined with data from the scorers table, including game

events like fouls and turnovers, the game and shot clock, the current score, and other game elements.

Figure 2.2: Representation of how SportVu cameras are positioned

According to the NBA, SportVu automatically collects a huge amount of statistics. These stats are combined into other hybrid player effectiveness stats as well. This information is provided to teams as reports and data visualizations [22, 21].

This leads to have available a lot of new type of statistics data, this remind to the Moneyball case [16, 18]. Moneyball is the story of the Oakland Athletics in the late 1990s and early 2000s, and how they, one of the poorest teams in the MLB, performed as well as the richest teams, re-evaluating the way a baseball how players are selected and engaged [14].

They wrote a book and also made a movie about this story.

The Athletics attempted to find statistical indicators of player performance, demonstrating that the previous method was not so efficient.

This allowed them to draft players than other teams overlooked, and get them for a reasonable price. This, in turn, allowed them to stretch the small budget further, and compete with times with double the amount of money [**moneyball** , 14].

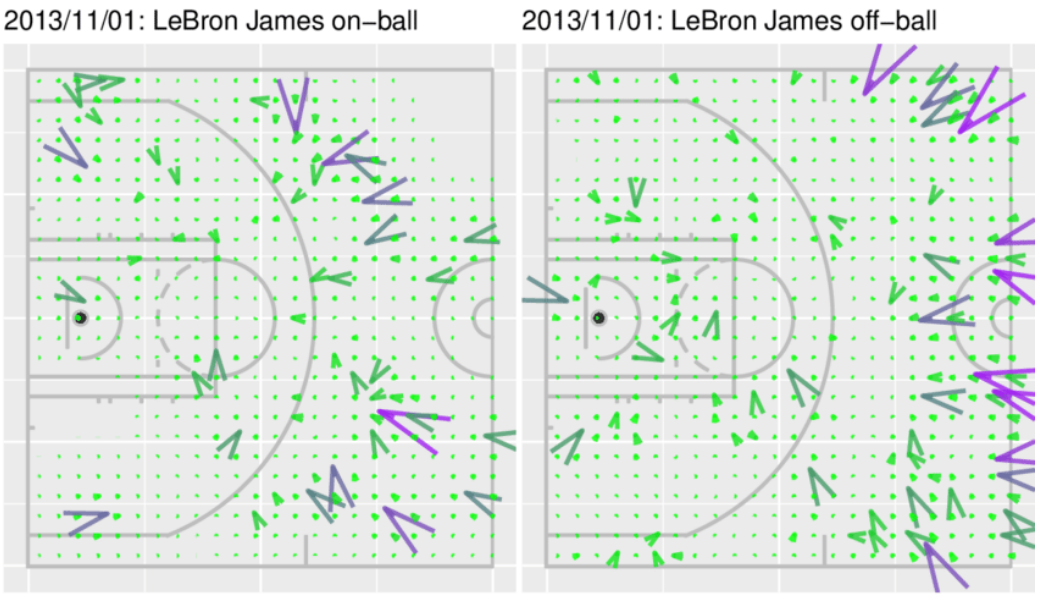

Figure 2.3: Visualization of smoothed empirical acceleration vectors for LeBron James for 2013/11/01 vs. the Brooklyn Nets

2.1.6 Volleyball

In volleyball a similar system was tested in 2014, at the FIVB Volleyball Women's Club World Championship in Zurich. It had six cameras placed around the court and connected to a computer [1].

Moreover in volleyball in all the high national matches a check video system is working in support to the referees to evaluate if the ball is "in" or "out" and for other actions. The video check can be called by each team in case of doubt. The system is called Videocheck, but it doesn't make statistics: it gives only support to the referee decisions.

It is possible to choose between the 19, 12 or 8 cameras System. The complete version of the Videocheck system is composed by 19 cameras with cameras positioned in this way (fig. 2.4):

- Line Cameras - 8+2 cams (1-8 and C-D)
- Net Cameras - 2 cams (A-B)
- Scene Camera - 1 cam (S)
- Block Touch - 6 cams (J-K-L-R-T-Y)

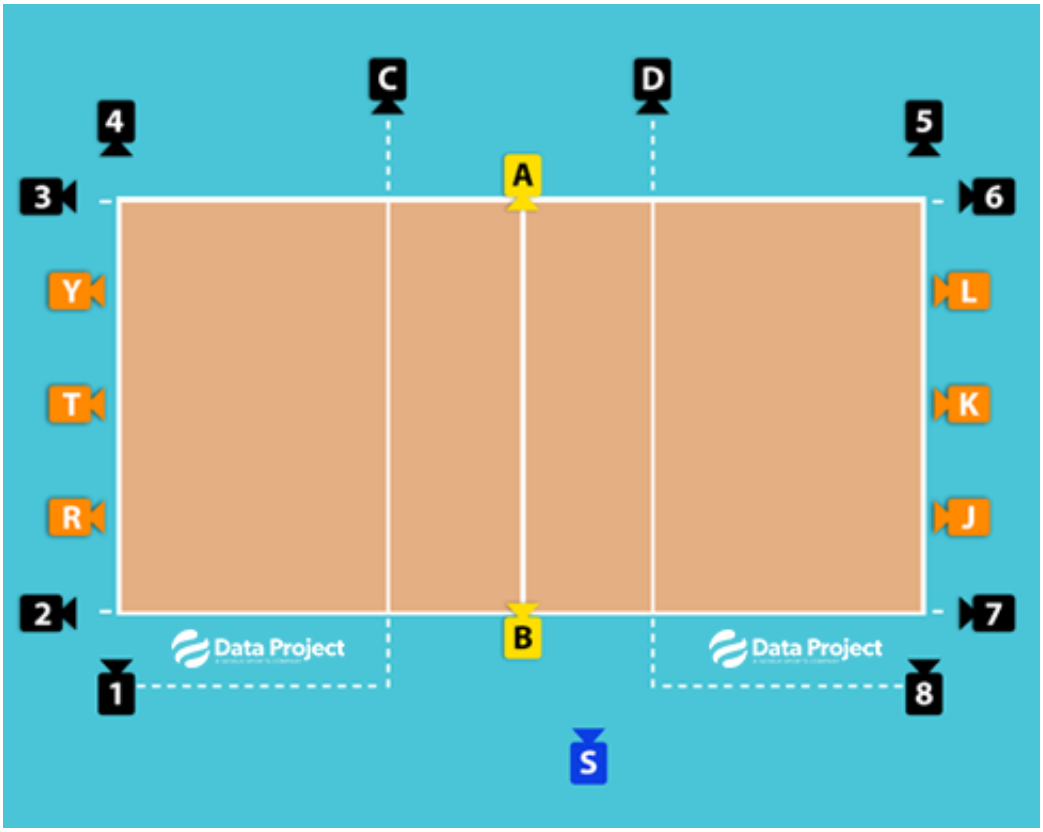

Figure 2.4: Representation of how Videocheck cameras are positioned

2.2 Challenges Of Computer Vision

Despite the great potential that computer vision, there are still critical challenges that need to be overcome, because human eyes are more trained in **recognizing posture** or solve **tracking problems** with occlusion players because other players in the court.

Tracking of sports players is also particularly challenging due to the fast and erratic motion, similar appearance of players in team sports, and often close interactions between players.

Also tracking the ball is a further challenge in team sports, where several players can occlude the ball (i.e. a ruck in Rugby Union), and it is possible that players are in possession of the ball with either their hands or between their feet.[4]

In general the problem is not related only to how the eye works, but in general it depends also to the experience that humans have in all the other topics in the real world and it's very difficult to replicate our inherited knowl-

edge of the world in a computational model, mostly because we associate a semantic meaning to everything around us.

Chapter 3

Background: Machine Learning and Computer Vision

3.1 Computer Vision

Computer vision is an interdisciplinary field that deals with how computers can gain high-level understanding from digital images or videos. The history and the evolution in this field is very long, but nowadays the most effective systems are models of particular artificial neural networks which try to optimize e discover the most important feature inside images exploiting the spatial information inside a image. The tasks in Computer Vision includes, concern extraction of high dimensional data from the real world in order to produce numerical or symbolic information, that can be meaningful and processed by a software or another system.

It can perform a variety of tasks in a wide range of fields, from self-driving cars to medical diagnosis.

That includes object or event detection, object tracking, pose estimation, face recognition, image restoration and everything concerns the analysis of images.

In Computer Vision we can have different task domains:

- `Image Classification`: given an image we expect a discrete label as output. We assume that there is only one object in the image.
- `Classification with Localization`: we expect not only "What" but also "Where" the object is.present in the image. The output of this further task is a numerical vector which define a Bounding Box area in the image.

- `Object Detection`: the image has not only one object, but can contain multiple objects. The task is to classify and localize all the objects in the image drawing several bounding boxes.

- `Semantic Segmentation`: the task is to label at the pixel level of an image with a corresponding class. The output is not more bounding boxes but is an high resolution image (typically of the same size of the input image).

- `Instance segmentation`: a semantic segmentation wherein along with pixel level classification, we expect the computer to classify each instance of a class separately.

Computer Vision aims also to replicate parts of the complexities in human vision system and visual perception. In the last years the trend that has dominated the field is the incorporation of machine learning techniques, that are more flexible in understanding the variability in the real world, because they can generalize better

3.2 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [7]

According to this definition the term "machine learning" refers to algorithms that learn through experience just how humans do, and a software model to learn to provide a desired output through experience needs data that describes historical states or in general examples to understand how it has to behave.

A data-driven approach in learning brings out a lot of well known problems related to the data, it is the core of autonomous processes in A.I.

Data often are biased, incomplete, noisy, not representative of the real world and this lead to train not perfect models. But this just the way used by our brain does, and despite and just for this imperfection it's represent a good way to generalize and interpret the surrounding world.

Not perfect but very close to that we define intelligence, and since we auto define ourselves intelligence, therefore whatever looks like us surely must be intelligence.

3.3 Deep Learning

Neural networks are one of the most popular and most famous machine learning techniques. They are based on a simplified model of an organic brain, composed of individual "units" that take in multiple real-valued inputs and output a single real-valued output.

These "units" are essentially perceptrons that evaluate a weighted linear combination of its inputs with an activation function that provide the non linearity to the function [17].

Units are arranged into layers that feed their outputs to each other. Ultimately, a final output row that dictates the outcome of feeding the data to the neural network, usually this is a classification.

The actual layout of the network, activation functions used, method to update the weights, and connections between the layers vary. Variants of neural networks exist that are better suited to images, recurrent data, and other data types [17].

3.4 Convolutional Neural Networks

CNN is an artificial neural network with an architecture inspired by the animal visual cortex. A Cnn network has convolutional layers (for feature extraction) and usually on the tail a dense fully connected layer (for the classification task).

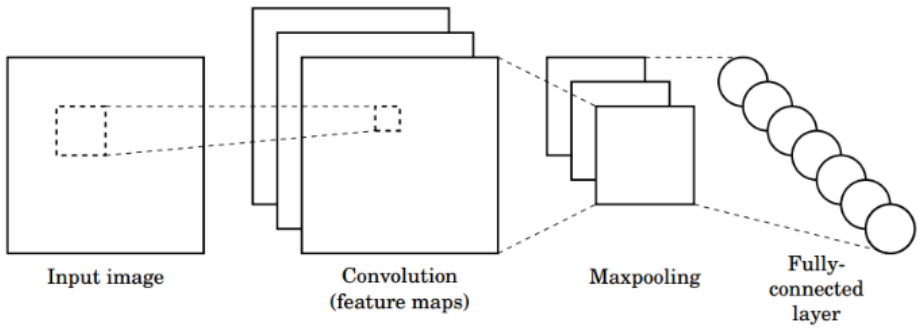

Figure 3.1: A Convolutional Neural Network model

The layers of CNNs exploit filters along the depth of a input (usually an image or a time series) extracting from a generic input information that

represent the features composition of every specific input data.

In fact applying specific filters (or kernels) sliding over the input the network extracts the so-called *feature maps*, that are the visual properties of each image.

IN 2012 a new paradigm model was proposed in ILSRVC competition named Alexnet which, instead to apply a fixed pre-computed representation of the features of the images, learns representation directly from the task combining different classifier layers using a non linear function (activation functions like Relu) that let to the model to learn the best representation by combining and sharing image attributes in a best way.

Indeed sharing attributes is useful to achieve the goal having a smaller spatial representation of features than the previous approach, avoiding the curse of dimensionality that exists for example a linear classifier like Knn that defines a new hyperplane for each new feature.

Using fully connected layers would make the model too precise and can lead to overfit it, moreover the parameters that we'll use, would be squared in the image size, therefore fc-layers are not efficient in computer vision tasks.

Convolutional layers instead apply kernels to the image to obtain detailed information from local areas by a cross correlation between neighboring pixels.

Convolutional operations can be interpreted as matrix multiplications: if we reshape inputs and outputs, the result matrix is a sparse matrix, which shares parameters across its rows, naturally adapts to varying input sizes, is equivariant to translations of the input, and since it can be used in parallel computation (gpu), is very efficient in memory and in time.

3.4.1 Receptive Fields

The most of computer vision task need not only large scale information but also detailed information from local areas, this is what the convolutional operations perform.

These operations lead to the definition of Receptive Fields on different levels of the network models.

Receptive fields are those pixels that affects the units of the next convolutional layer, they define the location and the size of the region in the input that produces the feature in a following layers.

A convolutional unit only depends on a local region (patch) of the input and each size grows linearly with the number of layers. On the contrary, in a fully connected layers, each unit has access to all the input region.

Not all pixels in a receptive field contribute equally to an output unit's response: pixels at the center of a receptive field have a much larger impact

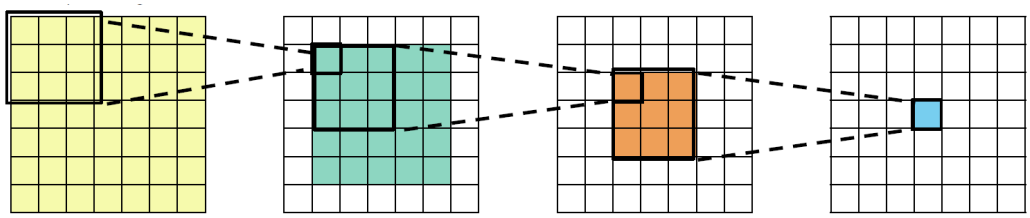

Figure 3.2: Receptive Fields over the layers

on output since they have more "paths" to contribute to the output.

The challenge for convolutional architectures is to design a model so that its receptive fields covers the entire relevant input image region saving resources.

Indeed to compute attributes corresponding to a large portion of the high resolution input, we would need too many layers (a very depth network) to let the final layer to cover the whole input. To obtain larger receptive fields with a limited number of layers, we downsample the activations inside the network or use other techniques like striding, pooling or dilated convolutions.

Pooling operations and dilated convolutions turn out to be effective ways to increase the receptive field size quickly.

Dilations introduce "holes" in a convolutional kernel: while the number of weights in the kernel is unchanged, the weights are no longer applied to spatially adjacent samples, but to a larger area.

Dilated convolutions make the receptive field grow exponentially and the number of parameters grow linearly.

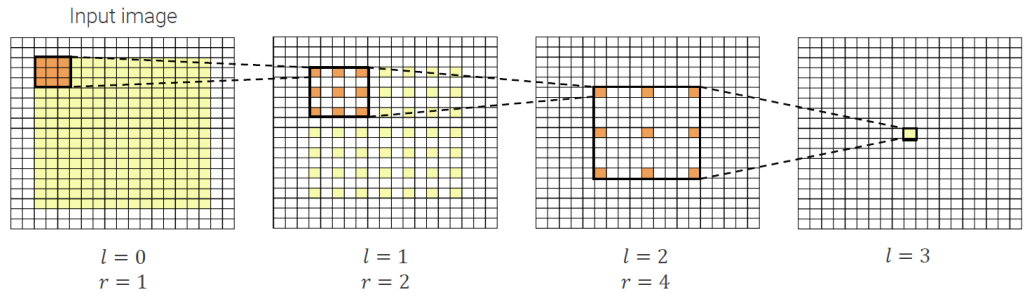

Figure 3.3: Dilated Convolutions over the layers

With Skip-residual blocks introduced with ResNets the network has Skip-connections and use different paths to preserve the information of the previous layer, therefore features can be learned with a large range of different receptive fields.

By introducing n skip-residual blocks, the networks utilize 2^n different paths and therefore features can be learned with a large range of different receptive fields

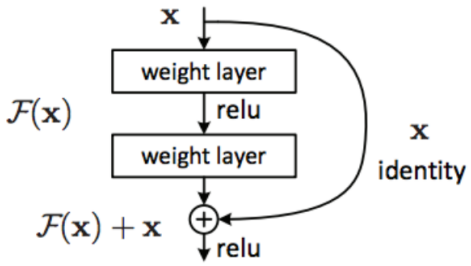

Figure 3.4: Resnet model

As models evolved the receptive fields increased (this is a natural consequence of the increased number of layers). In the most recent networks, the receptive field usually covers the entire input image: the context used by each feature in the final output feature map includes all of the input pixels

3.4.2 CNN models

Features Extraction from an image is the key for the most of computer vision task. To perform that, different models were proposed in the last years that let us to understand better the image properties:

- `Alexnet` the first modern feature extractor
- `VGG` repetition of regular stages. Each stage has same receptive field of larger convolutions but requires less params and computation, and introduces more non-linearities.
- `ResNet` Residual blocks and Bottleneck Residual blocks
- `InceptionNet` layers with different kernel size and StemLayer + GlobalAvgPool Classifier
- `DenseNet` shortcut connections.
- `DLA` hierarchical layer aggregation

Nowadays a lot of variants and new architectures are tested to improve more and more the capability of a system in specific tasks and different models are applied according to them.

ConvNet Model	Recept.Field	Effect.Stride	Effect.Padding	Year
alexnet_v2	195	32	64	2014
vgg_16	212	32	90	2014
mobilenet_v1	315	32	126	2017
mobilenet_v1_075	315	32	126	2017
resnet_v1_50	483	32	239	2015
inception_v2	699	32	318	2015
resnet_v1_101	1027	32	511	2015
inception_v3	1311	32	618	2015
resnet_v1_152	1507	32	751	2015
resnet_v1_200	1763	32	879	2015
inception_v4	2071	32	998	2016
inception_resnet_v2	3039	32	1482	2016

Source: `https://distill.pub/2019/computing-receptive-fields/`

3.4.3 MobileNet

MobileNet is a Convolutional neural network developed by google which is trained on the ImageNet dataset, majorly used for Image classification in categories and target estimation. It is a lightweight model which uses depthwise separable convolution to deepen the network and reduce parameters, computation cost, and increased accuracy. MobileNetV2 added an expansion layer in the block to get a system of expansion-filtering-compression called Inverted Residual Block

3.5 Object Detection

In Object Detection, Regional Proposal Network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal, This is very expensive in term of computational cost.

For that reason other models were developed hold like YOLO and Single Shot Detector (SSD) that detect multiple objects within the image just taking "Only Look Once" or in "one Single Shot", making the task much faster.

3.5.1 You only Look Once (YOLO)

Yolo treats detection as regression dilemma which takes an input image and learns the class possibilities with bounding box coordinates. It is based on fully connected network (FCN).

YOLOv3 uses a features extractor called Darknet and trained on ImageNet. It makes detections at three different scales (three different sizes at three different places in the network).

It divides every image into a grid of $S \times S$ and every grid predicts N bounding boxes and confidence. The confidence reflects the precision of the bounding box and whether the bounding box in point of fact contains an object in spite of the defined class. Then a classification score for every box for each class is given.

3.5.2 Single Shot Detector (SSD)

SSD runs a convolutional network on input image only one time and computes a feature map. Then run a small 3×3 sized convolutional kernel on this feature map to foresee the bounding boxes and categorization probability.

SSD also uses anchor boxes at a variety of aspect ratio comparable to Faster-RCNN and learns the off-set to a certain extent than learning the box. In order to hold the scale, SSD predicts bounding boxes after multiple convolutional layers.

3.5.3 PoseNet

Posenet[19] is a real-time pose detection technique with which you can detect human beings' poses in Image or Video. It works in both cases as single-mode (single human pose detection) and multi-pose detection (Multiple humans pose detection).

Posenet is a deep learning TensorFlow model that allows you o estimate human pose by detecting body parts such as elbows, hips, wrists, knees, ankles, and form a skeleton structure of your pose by joining these points.

It is trained with a MobileNet Backbone and its outputs is a pose, containing both a pose confidence score and an array of 17 keypoints.

Each keypoint contains a keypoint position and a keypoint confidence score. Again, all the keypoint positions have x and y coordinates in the input image space, and can be mapped directly onto the image.

Figure 3.5: the 17 keypoints detected by a Posenet

3.5.4 Centernet

CenterNet[23] is an anchorless object detection architecture. This structure has an important advantage in that it replaces the classical NMS (Non Maximum Suppression) at the post process, with a much more elegant algorithm, that is natural to the CNN flow. This mechanism enables a much faster inference.

Object area definition is crucial in Object Detections: most detectors use multiple basic boxes, or anchors, to encode their predictions by brute force methods. Each spatial cell in the output feature map predicts several boxes. Each box prediction is encoded as x- and y- offsets relative to the cell center, and width- and height-offsets relative to the corresponding anchor.

This leads to the generation of many garbage predictions: the post processing checks for overlapping predictions and the selection method to choose the right bounding boxes is not certain because the selection is made setting handcrafted thresholds.

If the overlap is too high (typically $IoU > 0.7$) then the predictions are assumed to refer to the same object, therefore is discarted. At the same manner, also if the prediction is too low is discarded.

This method is very expensive (also for a single object) because forces

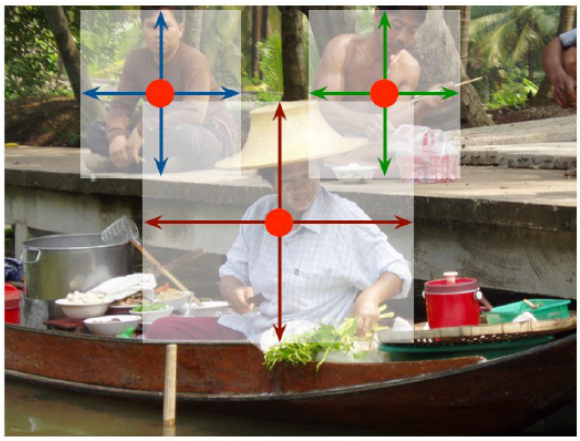

Figure 3.6: Centernet: Objects as points (center of the objects)

the network to decode all the predictions before the NMS, performing more time and power consuming operations on mostly irrelevant predictions.

Anchors are out Machine Learning processes: not differentiable and hard to train but they have a huge effect on the quality of the model.

The proposal by Zhou et al.[23] was to treat objects as points[1] and regress their sizes or other properties starting from that points, that represents the center of the object to detect. In this way the Objects detection task is now a Keypoints detection task.

Points are assigned based on location, and there aren't more huge garbage-predictions to select and discard by handcraft defined thresholds. There is only one positive "anchor" per object

At training-set preparation, they draw a map with delta functions at ground-truth box centers. A gaussian filter blur the centers, generating a smooth distribution that reaches a peak at object centers. The model uses two prediction heads: one trains to predict the confidence heat map, the other head trains, as in other detectors, to predict regression values for box dimensions and offsets, which refers to the box center predicted by the first head.

[1]Other model using similar strategy: CornerNet (detects 2 corners that define Bbs), ExtremeNet (2 corners + center). Both require a combinatorial grouping stage after keypoint detection, very slow. CenterNet extract points without the need for grouping or post processing.

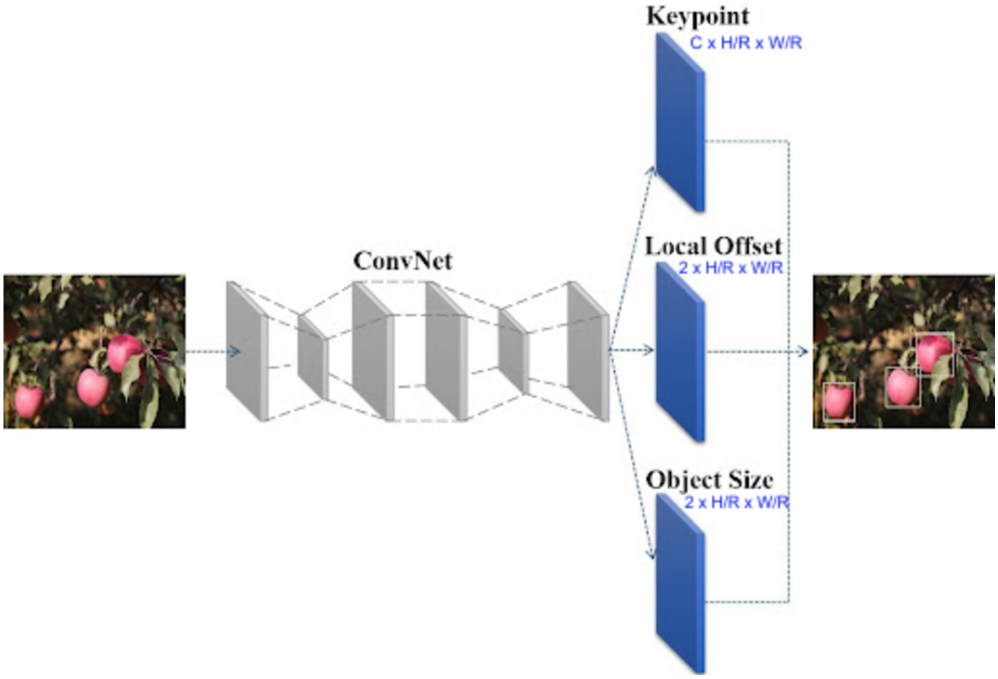

Figure 3.7: The Centernet model architecture

3.6 Mean Average Precision (mAP)

In computer vision, mAP is a popular evaluation metric used for object detection (i.e. localisation and classification). Localization determines the location of an instance (e.g. bounding box coordinates) and classification tells you what it is (e.g. a dog or cat).

In order to calculate mAP, first, you need to calculate AP per class.

Object detection systems make predictions in terms of a bounding box and a class label.

For each bounding box, we measure an overlap between the predicted bounding box and the ground truth bounding box. This is measured by IoU (intersection over union). For object detection tasks, we calculate Precision and Recall using IoU value for a given IoU threshold.

Precision measures how accurate your predictions are

$$PRECISION(p) = \frac{TP}{TP + FP}$$

TP = True Positives (Predicted as positive as was correct)
FP = False Positives (Predicted as positive but was incorrect)

That also means that for a prediction, we may get different binary TRUE or FALSE positives, by changing the IoU threshold. Recall measures how well you find all the right targets. The Recall for positive is:

$$RECALL(r) = \frac{TP}{TP + FN}$$

TP = True Positives (Predicted as positive as was correct)
FN = False Negatives (Failed to predict an object that was there)

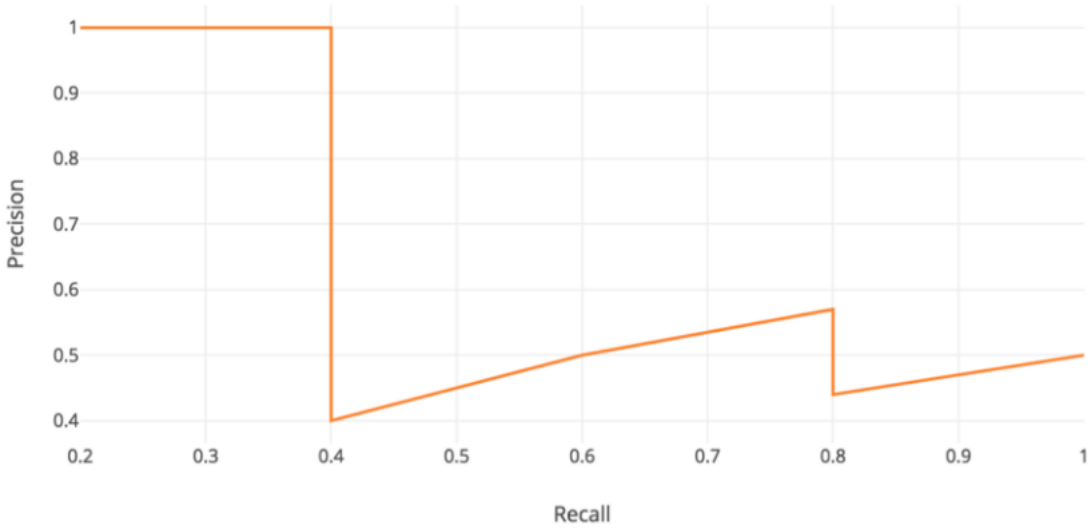

Figure 3.8: Precision Recall curve

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

$$AP = \int_0^1 p(r)dr$$

3.6.1 PASCAL VOC

For the PASCAL VOC challenge, a prediction is positive if $IoU \geq 0.5$. Also, if multiple detections of the same object are detected, it counts the first one as a positive while the rest as negatives.

In Pascal VOC, an average for the 11-point interpolated AP is calculated.

$$AP = \frac{1}{11} \sum_{r \in (0, 0.1, \ldots, 1)} p_{interp(r)}$$

The AP summarizes the shape of the precision-recall curve, and, in VOC 2007, it is defined as the mean of precision values at a set of 11 equally spaced recall levels [0,0.1,. . . ,1] (0 to 1 at step size of 0.1)

Mean Average Precision (mAP) is the averaged AP over all the object categories.

3.6.2 COCO mAP

Usually, as in VOC, a prediction with IoU ¿ 0.5 is considered as True Positive prediction. It means that two predictions of IoU 0.6 and 0.9 would have equal weightage. Thus, a certain threshold introduces a bias in the evaluation metric. One way to solve this problem is to use a range of IoU threshold values, and calculate mAP for each IoU, and take their average to get the final mAP.

$$mAP_{COCO} = \frac{mAP_{0.50} + mAP_{0.55} + ... + mAP_{0.95}}{10}$$

For COCO, AP is the average over multiple IoU (the minimum IoU to consider a positive match).

AP@[.5:.95] corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05. For the COCO competition, AP is the average over 10 IoU levels on 80 categories (AP@[.50:.05:.95]: start from 0.5 to 0.95 with a step size of 0.05).

3.6.3 COCO Keypoint Evaluation

Tresholding the IoU defines matches between the ground truth and predicted objects and allows computing precision-recall curves. Defining an analogous similarity measure to define an Object Keypoint Similarity (OKS) which plays the same role as the IoU, like AP/AR for keypoints detection [15].

COCO defines the Object Keypoint Similarity as:

$$OKS = \frac{\sum_i \exp\left(-d_i^2/2s^2k_i^2\right)\delta\left(v_i > 0\right)}{\sum_i \delta\left(v_i > 0\right)}$$

Where d_i is the Euclidean distance between the detected keypoint and the corresponding ground truth, v_i is the visibility flag of the ground truth (the detector's predicted v_i are not used), s is the object scale, and k_i is a per-keypoint constant that controls falloff.

To compute OKS, we pass the δ through an unnormalized Guassian with standard deviation sk_i

For each keypoint this yields a keypoint similarity that ranges between 0 and 1. These similarities are averaged over all labeled keypoints (keypoints for which $v_i > 0$). Predicted keypoints that are not labeled ($v_i = 0$) do not affect the OKS.

Perfect predictions will have OKS = 1 and predictions for which all keypoints are off by more than a few standard deviations si will have $OKS \sim 0$.

The OKS is analogous to the IoU. Given the OKS, we can compute AP and AR just as the IoU allows us to compute these metrics for box/segment detection [15].

Chapter 4

The Project

The project was developed with the Gluon framework that provides pre-trained models for object detection and pose estimation and other useful functionalities in machine learning for computer vision.

We used also a public labeled datasetof volleyball images released by Mostafa S. Ibrahim et al.[12, 13] to get target information about the type of player action in the images.

We also use that dataset as base to augment the description of the data: we manually labeled court shape information in several frames to train our custom centernet network from scratch, starting from the centernet model used for pose estimation[2, 23].

4.1 COCO Dataset

The COCO dataset is a large-scale dataset for many computer vision tasks. The main features: 1.5 million object instances, 80 categories of objects, 250,000 people poses, Object segmentation, 330K images (>200K labeled)

COCO dataset is composed of image files and annotation files. The annotation file is a JSON containing all metadata about the class of the target objects in the images, category (or subcategories), position and size of bounding boxes, area, polygons for segmentation, keypoints, file details of source images

The COCO dataset format

For our test we used the images in train2017.zip and val2017.zip and the annotations in annotations_trainval2017.zip

```
{
    "info": {...},
    "licenses": [...],
```

```
        "images": [...],
        "annotations": [...],
        "categories": [...],
        "segment_info": [...]
    }
```

annotation section The *annotation* section contains all the information about the position, the segmentation and the keypoints of the objects inside the images. The elements in the set of annotations can contain *bbox* data, *segmentation* data (which are a list of polygon vertices), *keypoints* data (a set of coordinates).

There is one annotation for each object and one or all the three type of data can be.

```
"annotations":
[
    {   "segmentation": {"counts":"....","size": [ width, height ]},
        "image_id": image_id,
        "bbox":[x, y, w, h],
        "num_keypoints": keypoints_nr,
        "area": int,
        "iscrowd":  0|1 ,
        "keypoints":[x1,y1,c1,x2,y2,c2,..., xk,yk,ck],
        "id": annotation_id,
        "category_id": cat_id
    },...

]
```

contains the body part locations and detection confidence formatted as x1,y1,c1,x2,y2,c2,.... c is the confidence score.

Bounding boxes have a special flag *iscrowd* to determine if the content should be treated as a crowd (no keypoints) or a person (should have keypoints).

In general, iscrowd is set for a bounding box that contains many small instances of people, such as an audience at a tennis match

categories section

```
"categories":
[
   {
    "supercategory": "person",
    "id": 1,
    "name": "person",
    "keypoints": ["nose","left_eye","right_eye","left_ear","right_ear","left_shoulder",
    "right_shoulder","left_elbow","right_elbow","left_wrist","right_wrist","left_hip",
    "right_hip","left_knee","right_knee","left_ankle","right_ankle"],

    "skeleton": [[16,14],[14,12],[17,15],[15,13],[12,13],[6,12],[7,13],[6,7],
    [6,8],[7,9],[8,10],[9,11],[2,3],[1,2],[1,3],[2,4],[3,5],[4,6],[5,7]]
    }

]
```

4.2 GLUON package

Gluon [9] package is a high-level interface for MXNet designed to be easy to use while keeping most of the flexibility of low level API.
GluonCV [10] is a Deep Learning Toolkit for Computer Vision that provides implementations of state-of-the-art (SOTA) deep learning algorithms in computer visions.

GluoCV aims to help engineers, researchers, and students quickly prototype products, validate new ideas and learn computer vision, making available some libraries that let to test or train models very easily, and provides also several pre-trained network models for different tasks.

In particular for our project we used 2 pre-trained models for the object detection task (players and ball) and for the pose estimation task

For the detection task we used a *yolo3_mobilenet1.0* trained on COCO model

For the pose estimation task we used a *alpha_pose_resnet101* trained on COCO model in Gluon you can reset all the classes you don't need, so for the object detection we kept only "person" and "sports_ball"

```
import mxnet as mx
from gluoncv import model_zoo, data, utils
from gluoncv.data.transforms.pose import detector_to_alpha_pose,
                heatmap_to_coord_alpha_pose

ctx = mx.gpu(0)
detector  = model_zoo.get_model('yolo3_mobilenet1.0_coco',
            pretrained=True, ctx=ctx)
pose_net = model_zoo.get_model('alpha_pose_resnet101_v1b_coco',
            pretrained=True, ctx=ctx)

detector.reset_class(["person","sports_ball"],
            reuse_weights=["person","sports_ball"])
```

4.3 Volleyball Dataset

For the project we used a public dataset that contains several sequences of several volleyball matches of international women/men competitions. We'll refer to this dataset also with the name of "original dataset".

The dataset was released by Mostafa S. Ibrahim et al.[12, 13] as support for a paper describing a deep temporal model for the group activity recognition problem from volleyball plays.

They annotated 4.830 frames from 55 videos with 9 player action labels and 8 team activity labels. For each annotated frame, other 40 frames (20 before and 20 after) are in the dataset, therefore in total there are 198.030 frames (around 60GB).

All the frames are in front of the court, from the long side of the game court.

The possible labels for player action are: *Waiting,Setting, Digging, Falling, Spiking, Blocking, Jumping, Moving, Standing* [1]

The dataset is divided in 55 different folders, one for match numbered from 0 to 54 [2].

Each main folder contains an annotation.txt file and other folders with 41 images inside. The 41 images inside are 41 frames that compose a video sequence of an volleyball action of the match of the parent folder (0..54).

```
- 0
  | - annotations.txt  # one for all the subfolders in 0
  | - 1000 -
        | - 980.jpg
        | - 981.jpg
        | ...
        | - 1000.jpg
        | ...
        | - 1019.jpg
        | - 1020.jpg
  .....
- 1
  | - annotations.txt  # one for all the subfolders in 1
  .....
...
- 54
```

Only the central frame contains the labels of the type of action for all the players.

The annotations are in the annotations.txt files, one for each match (0..54).

Each row contains a flat list of elements, the first one is the number of frame in the middle, which is also the name of the folder and the name of the jpg file inside. Like in the example here below in the annotation file of folder "/0" if a row contains 24425.jpg as first element, it means that the full path of the middle frame is /0/24425/24425.jpg and the other 40 files that make up the video sequence in the subfolder, will be named from 24405.jpg to 24445.jpg

Here below there is an example of the data inside:

```
...
24425.jpg r-pass 945 423 74 178 moving 889 332 75 125 digging
865 323 82 106 standing 678 315 68 127 moving 578 402 57 131 standing
556 443 120 151 standing 430 475 75 173 moving 469 287 43 134 moving
412 342 57 131 standing     381 408 65 130 standing
297 324 44 150 standing 151 337 57 125 standing
```

[1] We remind to the appendix 6.2 for the basic rules and player actions definition of volleyball.

[2] The complete list of the associations between folder and competition is in Appendix section 6.1.

```
...
18756.jpg r_spike 203 473 74 149 digging 203 405 61 128 digging
471 350 53 117 standing 717 387 49 197 blocking 707 352 56 183 blocking
783 472 75 105 waiting 883 430 88 152 spiking 918 409 63 121 standing
963 423 94 157 standing 1023 396 60 111 digging
...
```

The first element in each row is the file name of the middle frame, the second one is a label of the main action of the sequence and after a there is a series of couple of values composed by bounding box definition (a set of 4 numbers) and the label for each of them. Each couple in the format *X Y W H label* defines the bounding box of each player and its action

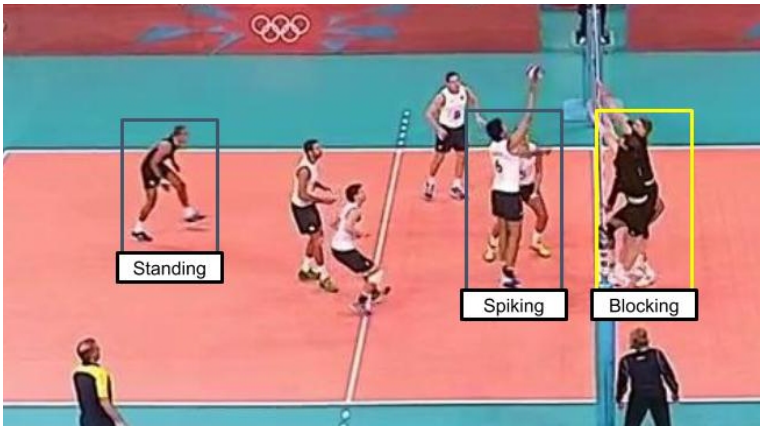

Figure 4.1: example of labels for the middle frame in each sequence

We also use dataset to create new annotations: we manually labeled court shape information in several frames to train our custom Centernet network from scratch, starting from the Centernet model used for pose estimation [2, 23].

4.4 Tasks architecture

To reach our goal we have to go through several steps:

- detection of the ball
- detection of the players
- labeling all the players in our detection, migrating the labels from the original dataset
- detection of the court area

- filtering players from all the people detected thanks to the court area
- tracking the players along the frames
- detection of the poses of the players
- calculate the homography matrix
- positioning the players on 2D representation of the court with associated the action type

4.5 Labels migration

Since our aim is to have coordinates of the gesture of each player along the time, we had to extend the existing annotations of the type action (currently in one only frame over 41), applying a object detection model to migrate the labels that we already had, over all the other frames.

4.5.1 Players Detection

First of all we performed a detection task for the ball and the persons on the image.

Before the detection we have to pre-process the image for the detector with preset data transforms. Here we specify that we resize the short edge of the image to 512 px.

```
x, img = data.transforms.presets.yolo.load_test(
                    frame, short=512)
```

This function returns two results:
- the first is a NDArray with shape (batch_size, RGB_channels, height, width). It can be fed into the model directly.
- the second one contains the images in numpy format to easy plotting. Since we only loaded a single image, the first dimension of x is 1.

At this point we can run the detector:

```
class_IDs, scores, bounding_boxs = detector(x)
```

And we got 3 arrays:
- *class_IDs*: the class of the detected objects (1 for person, 37 for sports ball)
- *scores*: the score assigned to the detection
- *bounding_boxs*: the bounding box coordinates of the object in the image

In this way, we got bounding boxes of all the players defined with our Gluon model. The first problem was to perform the migration of labels from

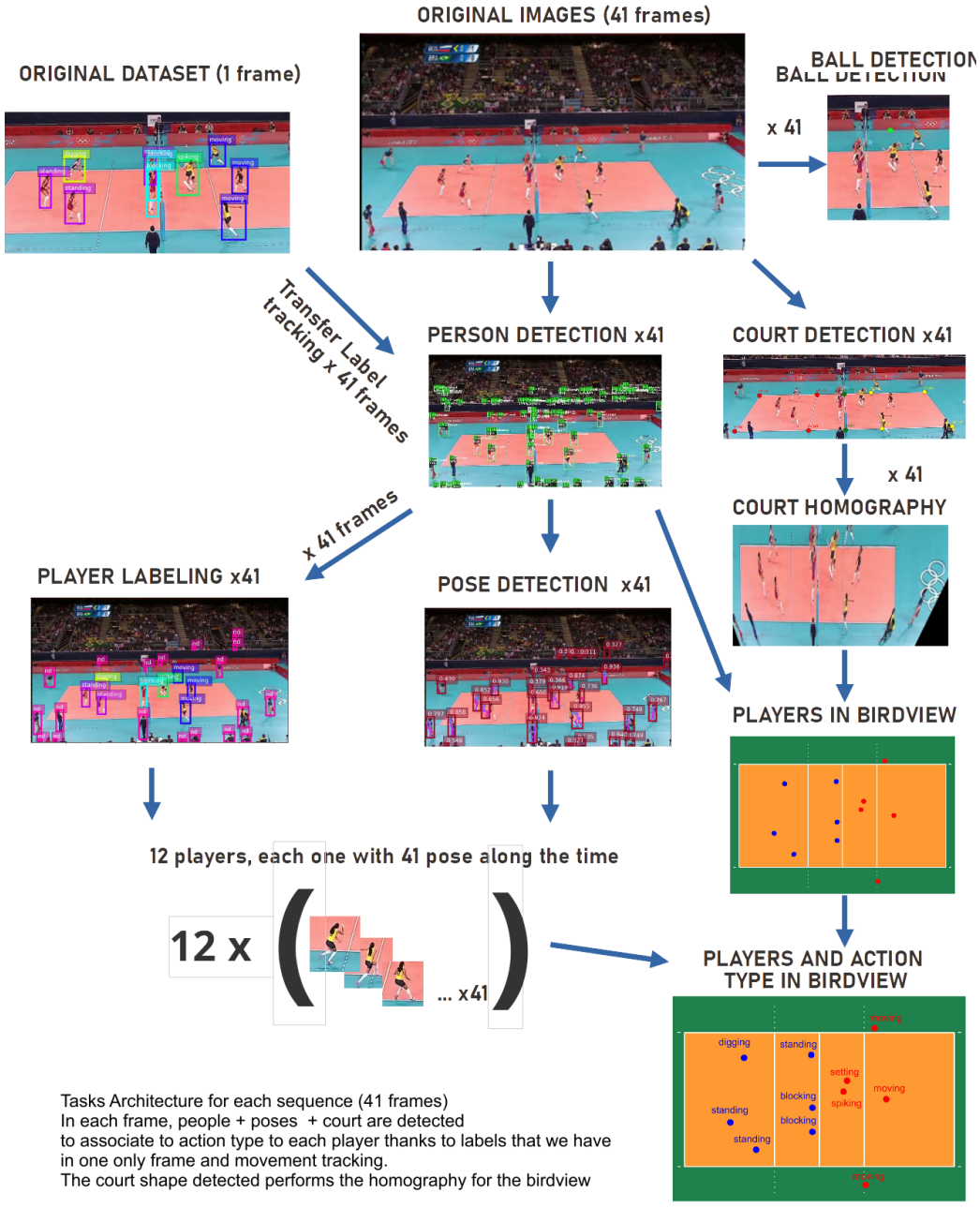

Figure 4.2: The workflow of the project tasks

the original dataset where the bounding boxes were manually set to perform a training task.

To match detected bounding boxes with ground-truth ones, we used the Intersection Over Union (IOU) value over all the bounding boxes in the two images of the middle frame (the once with labels), therefore we obtained labels on our bounding boxes without having to manually annotate them.

4.5.2 Players Tracking

Since we had the bounding boxes for all the other 40 frames of the sequence, we needed now to implement a tracking method which preserved the correspondence of the same player between one frame and another to assign on the unlabeled frames, the label of the action type that we have in the original dataset

In the beginning, to perform the tracking task, we tried to use simply the IOU, but it didn't work well. The main issue was that the players move during the play, of course. This cause problems of occlusions and also the position can reverse when two players cross each other.

To solve this issue we implemented a simple tracking algorithm, who kept memory of the players' movements along the frames.

Starting from the middle frame (the one with labels), we sliding frame by frame before on left direction (over 20 frames before) and then on right direction (for the 20 frames after). At each slide of 1 frame, we searched for the bounding box in that frame having a correspondence with the previous labeled frame to put over the same player the action type he was performing, to have in the end the same label for the same player in every frames of the entire sequence.

Since it's quite normal that during the play two players cross each other, and when it happens the bounding box of the occluded player disappear, we needed a method we to deal with this problem. So we implemented a sort of memory of movements for each player, by calculating the position (the center of their bounding boxes) for each frame, keeping stored them in an array that we used at every new sliding step to predict the position in the next frames.

To find a match with a bounding box at the next frame, we calculated IOU based on the prediction of the new position (not on that previous detected position).

In case of occlusion the array is preserved for next steps: for all the bounding boxes not found anymore, we search in previous frames the values of the ghost bounding boxes and copy the predicted values until it doesn't appear in the detection again.

Despite it's a very simple tacking system, this method perform very well also in absence of occlusion because if a player is moving in a direction, is more likely find her where the speed of the direction predict her position.

4.6 Poses detection

To perform this task, we process the tensor used for the player detection and use also its outputs (*class_IDs, scores, bounding_boxs*).

Alpha Pose network, which is our pose detector, expects the input has the size 256x192, and the human is centered. So we need to crop the bounding boxed area for each human, and resize it to 256x192, then finally normalize it.

In order to make sure the bounding box has included the entire person, we slightly upscale the box size.

```
pose_input, upscale_bbox = detector_to_alpha_pose(img,
                class_IDs, scores, bounding_boxs)
```

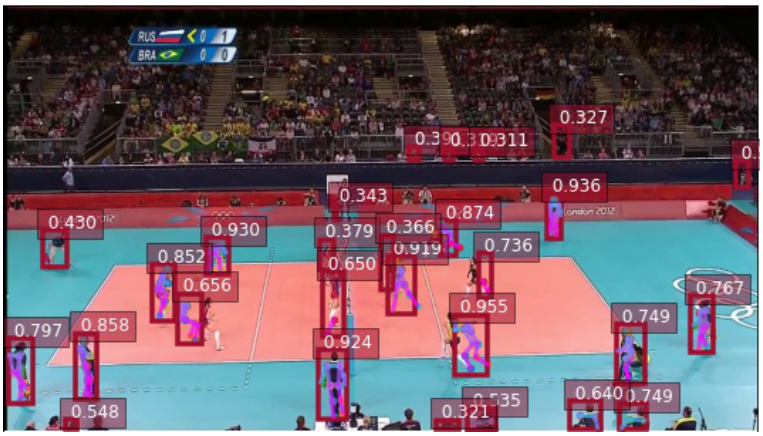

Figure 4.3: The pose detection in the volleyball court

An Alpha Pose network predicts the heatmap for each joint (keypoint). After the inference the model searches for the highest value in the heatmap and map it to the coordinates on the original image.

```
predicted_heatmap = pose_net(pose_input)
pred_coords, confidence = heatmap_to_coord_alpha_pose(
                    predicted_heatmap, upscale_bbox)
```

To display the pose estimation results it's possible to use a Gluon function

```
ax = utils.viz.plot_keypoints(img, pred_coords, confidence,
                    class_IDs, bounding_boxs, scores,
                    box_thresh=0.5, keypoint_thresh=0.2)
plt.show()
```

The skeleton labels that would be drawn on the detected human. Each joint is mapped to a unique number. This is the complete mapping of the 17 keypoints in AlphaPose [3]:

```
{0, "Nose"}, {1, "LEye"}, {2, "REye"}, {3, "LEar"}, {4 REar"},
{5, "LShoulder"}, {6, "RShoulder"}, {7, "LElbow"}, {8 "RElbow"},
{9 LWrist"}, {10, "RWrist"}, {11, "LHip"}, {12, "RHip"},
{13, "LKnee"}, {14, "Rknee"}, {15, "LAnkle"}, {16, "RAnkle"}
```

The pose estimation model has great dependency on the resolution of the camera.

If the player or the camera moves too fast, the player shown in the video might not be clear enough for the model to detect humans and label the pose skeletons accurately and precisely. And also in case of occlusion for sure you can't get any poses

As well as the occlusions, also for the poses, if we cannot get the pose in a frame, we copy the closer player's pose in the time (the first we find, before or after).

Unlike the occlusion in the previous step and thanks to it, the lack of poses, due to the occlusion or other, is easier to solve because we already have the player identification along the frame.

We decided to copy the values from the closer frame because, instead to have empty vectors along the time, we think that it's better having a kind of still image in a training task process.

4.7 Court detection

This is an important part of the project, because we were in front of a big issue: we needed to find the exactly position of the court to calculate the homography matrix and can draw the bird's eye view of the players' position.

Moreover the shape of the court let us to filter a lot of garbage bounding boxes founded during the person detection step, saving time in the IOU search for tracking task and also to define important rules to filter people very close to the court and who easily can be confused with players (i.e. coaches, referees, ball boys)

4.7.1 Related work

The court is usually detected by classic computer vision algorithms, but it should be correctly detected even when the camera moves from a side to the other of the court, when camera zooms.

[3] See also the Fig. 3.5 for the graphical view

Certain videos can be more challenging for court detection than others: for example, when the court is partially out of the frame, when the brightness is either very low or very high or when the player occludes a large portion of a court line (it happens often in volleyball matches).

It's possible find several work about sports court estimation and the most of them uses at the beginning a Gaussian blur on the black/white image to reduce on the noises and Canny edge detection to find the edges. Finally a Hough line transform [5] is applied to find intersections of white lines.

Also we tried to use this method and results are quite good but not so stable, because were strictly depending on the brightness conditions and other factors.

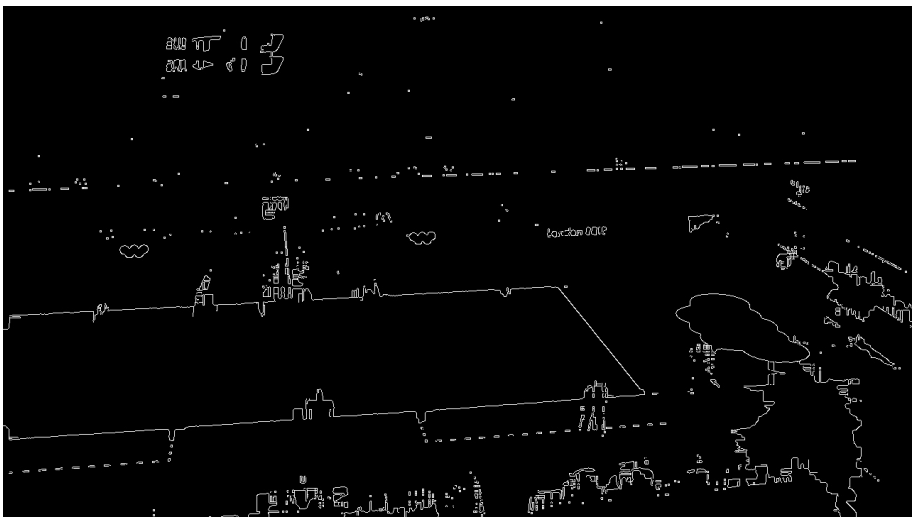

Figure 4.4: Canny alg. applied to the volleyball court

Another way is to exploit different colors between inside and outside the game court. But as the previous one, also this methods is very dependent on visual elements of the each court, so every time it would be necessary calibrate the camera or the algorithm in any new site.

4.7.2 "Courts as Points"

The solution that we propose exploits a recent model in object detection called Centernets (see 3.5.4).

Since Centernets are very effective at find keypoints, in particular human pose, we tried to build a Centernet that instead searching for human joints, searches for line joints of a volleyball court.

Starting from the code, written for detecting human joints using centernets in mxnet [2], which is a porting of the "Object as Points" [23], we modified the architecture of the regressor head (that used to find the 17 joints) to find, instead, 10 keypoints over images containing a volleyball match scene.

After customizing the model for our needs, to perform the training we needed an adequate dataset. The best solution was to re-use the images into the original dataset that already we used for the detection task.

Therefore, we manually labeled 500 different images, selecting them from only 12 different match-folders (0-9, 44, 45) over 54 available.

For each images we targeted 10 keypoints, corresponding to the joints of the court lines.

We adopted the COCO dataset format (see 4.1) to collect our data, so following its schema we needed to have a set of images in the *images* section, a set of labels in *annotations* section.

In the *categories* section we defined our custom model in the *keypoints* section, reporting the name of all our keypoints, numbering them from 0 to 9 starting from the top-left corner.

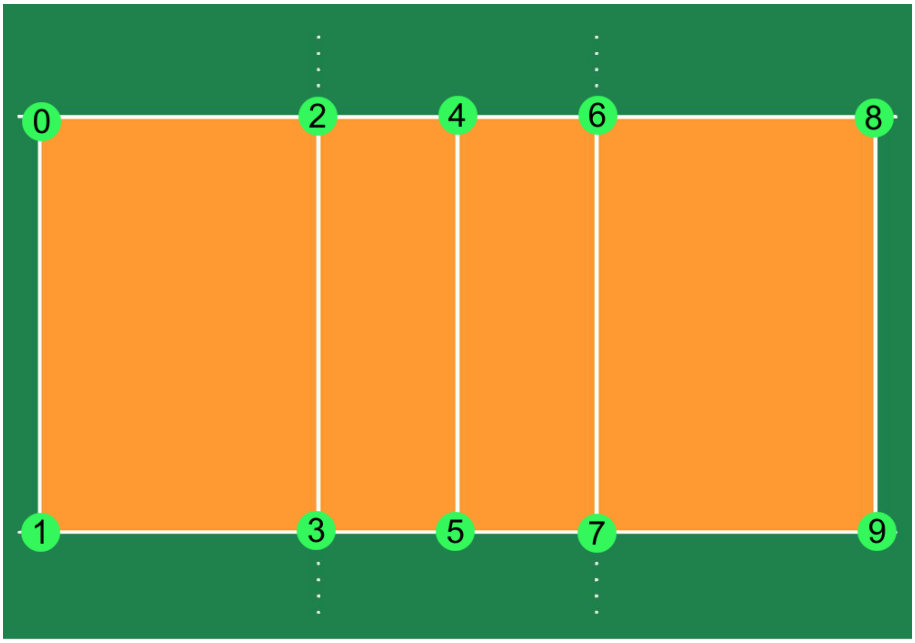

Figure 4.5: Indexes for the 10 keypoints

We defined also the *skeleton* of the joints of volleyball court, like we defined it. The skeleton definition is just the list of the connections between 2 vertexes, for example in the human skeleton schema, some of them represent the limbs.

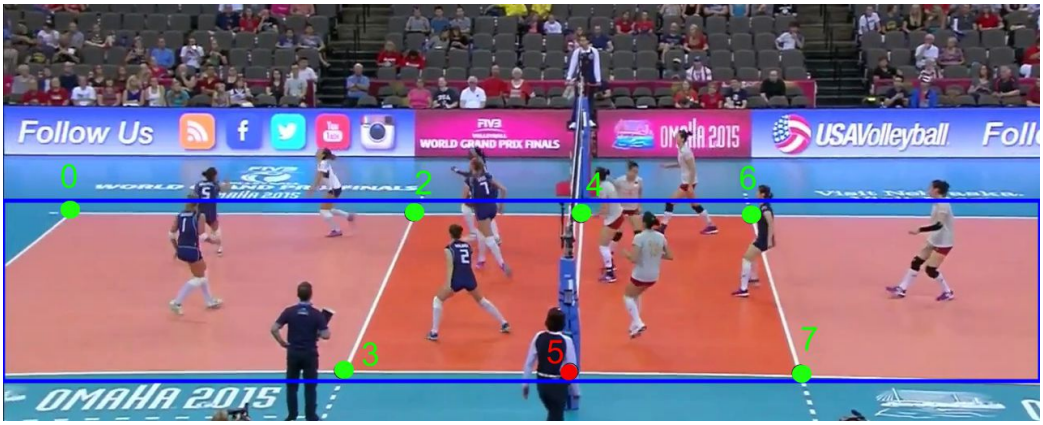

Figure 4.6: How we labeled our dataset


```
,"categories":[{"supercategory":"game_court","id":1,"name":"game_court",
"keypoints":["left-top","left-down","left-middle-top","left-middle-bottom",
"center-top","center-bottom","right-middle-top","right-middle-bottom",
"right-top","right-bottom"],
"skeleton":[[0,1],[0,2],[1,3],[2,3],[2,4],[3,5],[4,5],[4,6],
[5,7],[6,7],[6,8],[7,9],[8,9]]}]}
```


To manually target all the 500 photos we used a simple tool, then we build ourselves another tool to refine the tagging: we tracked also the information depending on the type of visibility of every keypoints in the frame: For each photo we annotated the coordinates of each 10 keypoints and also if they are visible, occluded in the scene (by a player) or out of frame, according to these indexes: 0:None, 1:Occluded, 2:Visible.

In fact, the COCO dataset format provides that the keypoints definition is a set of triplets (xk, yk, ck). xk, yk are the coordinates of the point, the ck is a point confidence parameter whose value has different meanings depending on the dataset. We didn't use it in the training process but we collected and annotated it anyway, mostly to identify how many keypoints have a $ck > 0$ and calculate the number keypoints to assign in the param "num_keypoints" of the dataset as COCO dataset format requires.

Again, this third parameters can be useful in future training process works

For each items in the dataset, we defined also *bbox* value, which define the rectangle circumscribed to our volleyball court (in blue in the figure).

After the tagging, we calculate that simply getting the min top/left and the max bottom/right values. This another element that the model predict together to the 10 keypoints and can be very useful.


```
...,
{
    "image_id":8027014,
    "bbox":[0,405,1280,222],
```

```
    "num_keypoints":7,
    "keypoints":[83,415,2,0,0,0,506,420,2,419,614,2,
                 713,422,2,700,615,1,923,423,2,
                 984,617,2,0,0,0,0,0,0],
    "id":8027014,
    "category_id":1
},
...
```


Here is an example of how our keypoints annotation was made for each image, according to the COCO format.

As you can see also in Fig. 4.6, 3 keypoints are Not Visible (index: 1, 8, 9) and 1 keypoint is Occluded (index: 5). In fact, the JSON assigns:
- (0, 0, 0) for the keypoint 1, 8, 9
- (700, 615, 1) for the keypoint 5
- (xk, yk, 2) for all the other visible keypoints

4.7.3 Training

After the definition of our court detector, customized the regression head of the centernet, now we had also the dataset in COCO format, ready to be used on our training model.

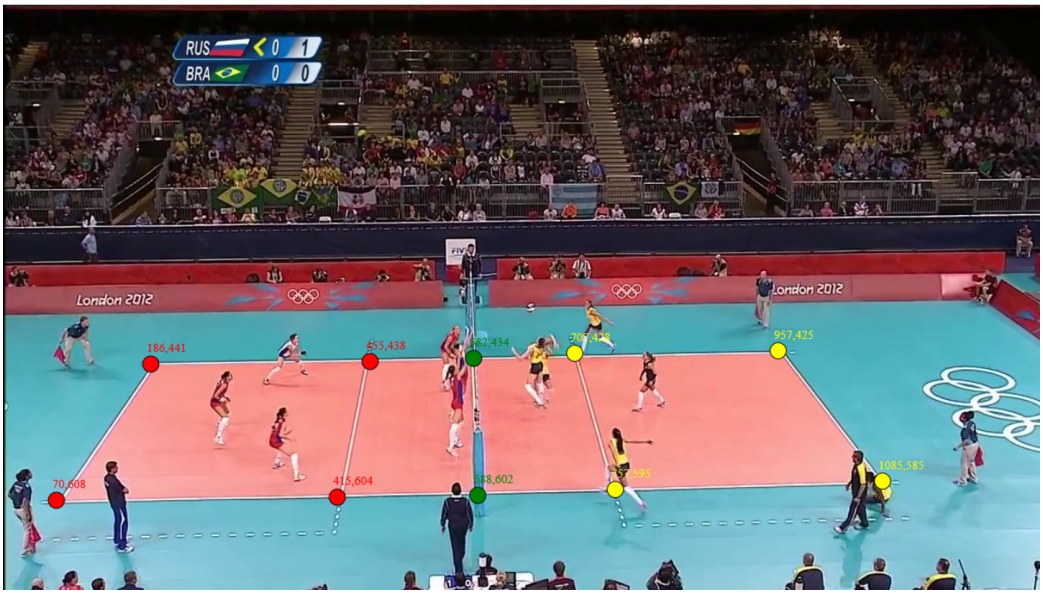

Figure 4.7: The output of the 10 court keypoints by our detector

We had 497 ground truth frames (it means about 4.970 points) and we divided them in train set of 430, and validation set of 67.

We trained set a CenterNet architecture with resnet18 network for 2D object detection.

We run the training for 700 epochs, using a NVIDIA RTX 2080 it takes less than 2 hours.

At the end of the training, our dectector were able to detect all the keypoints of the game court in a volleyball-match image.

4.7.4 Results

The results are quite impressive, because applying our detector over all the other 43 match-folders, that the detector didn't ever see before, the keypoints are found very well. It works in different conditions: different brightness, different camera zoom or point of view. It looks very robust in front of a lot of noisy conditions.

The interesting thing of our new detector is that it can retrieve not only the keypoints in the frame, but also those not in the scene. This means that we have the prediction also of the ghost keypoints, with negative coordinates or coordinates overflowing the size of the frame.

Having good points of reference for detecting the court is very important, because the court area, helps other parts of a complex system like this (filtering for example in detection task other people), and also lets to calculate some other interesting statistics, that before we can't (distance between players or dense maps on the court drawn from each player).

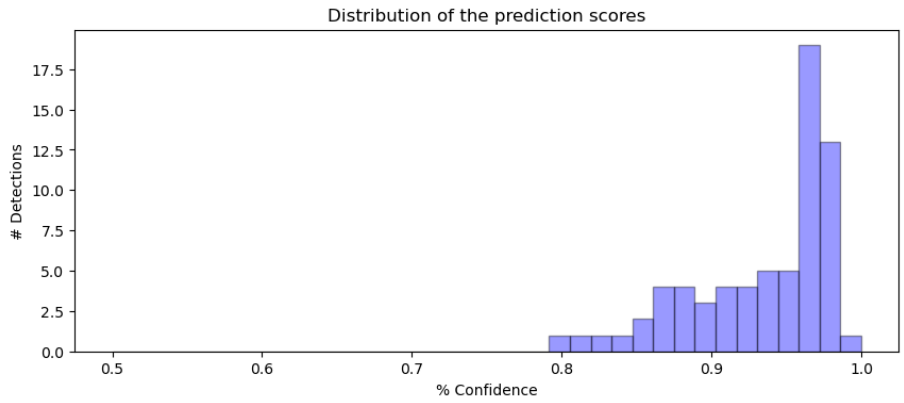

Figure 4.8: Distribution of the confidence scores

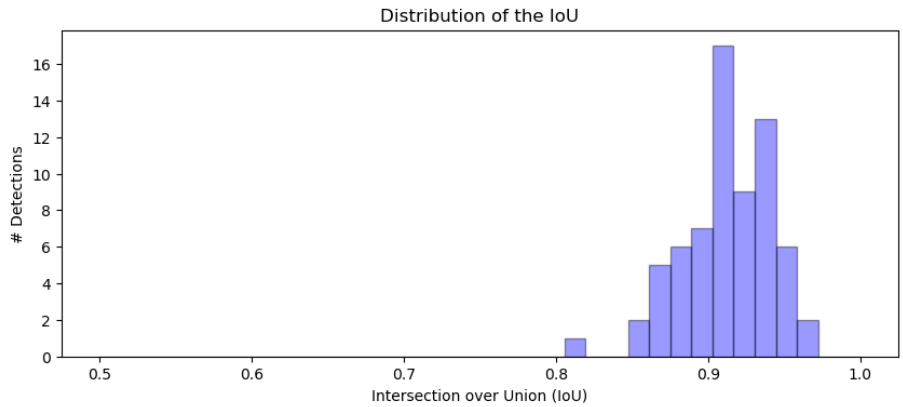

Figure 4.9: Distribution of the IOU values

4.7.5 Evaluation

It's not easy to understand how to find a metric to evaluate our task: in the end, this is an object detection task or better an object segmentation task. We could evaluate it by using the COCO mAP (see section 3.6 about mAP).

We didn't have a test dataset, we performed it over our validation dataset composed of 68 frames.

In the prediction task, the average of the score confidence of the bounding box (and the points) and is 0.93% in all the cases except one $> 0.8\%$ (see histogram fig.4.8).

We considered both the Ground Truth shape and the prediction shape, calculating the area defined by only court points present in the frame scene ($vk \neq 0$).

Also the average of the IoU between the Ground Truth and the predicted court is 0.91% in all the cases $> 0.8\%$ (see histogram fig.4.9).

With this dataset and our detector, the Recall is always equal to 1. The calculation of the COCO mAP is:

mAP[.5]=1, mAP[.55]=1, mAP[.65]=1, mAP[.7]=1, mAP[.75]=1, mAP[.8]=1, mAP[.85]=67/68, mAP[.9]=48/68, mAP[.95]=6/68

The COCO mAP = 0.87794

4.8 Bird view making

Having the position in the image of all the joints (most of them) of the game court, we find the homography matrix quite easily, because we know very well the game court dimensions and the relative distances of all the court points

in the reality.

We can use the opencv function cv2.findHomography that does it for us, but it requires at least 4 correspondent points between the 2 images to calculate the right matrix. At least 4, but we can put inside the 2 arrays (src_pts, dst_pts) how many points we want to be more accurate.

Even if our detector find ghost keypoints out the scene, since we can't be sure about their prediction (neither the network could be of it, since it hasn't any point of reference on it), during the Homography Matrix calculation, we kept only all the keypoints that are inside the frame, and exclude those that are outside. We don't lack of keypoints for this task, because we have available usually 8 or 9 keypoints over 10 in most of frames and in the worst of the cases (zoom or close view on one side of the court) we have always 6 or more keypoints.

So after filtering the two lists, we can get the homographymatrix (H).

```
H, mask = cv2.findHomography( src_pts , dst_pts , cv2.RANSAC,5.0)
```

src_pts are the points in the image, dst_pts instead are the points of a fixed 2D court image. The size of the two arrays has to be the same, so every time we arrange the two array according to how many point are visible in the sec_pts

Now that we have our homography matrix (H), one for each frame, we can see how the whole image is projected in the 2D image model (see Fig.4.14) or add all the players' positions, by using another function of Opencv.

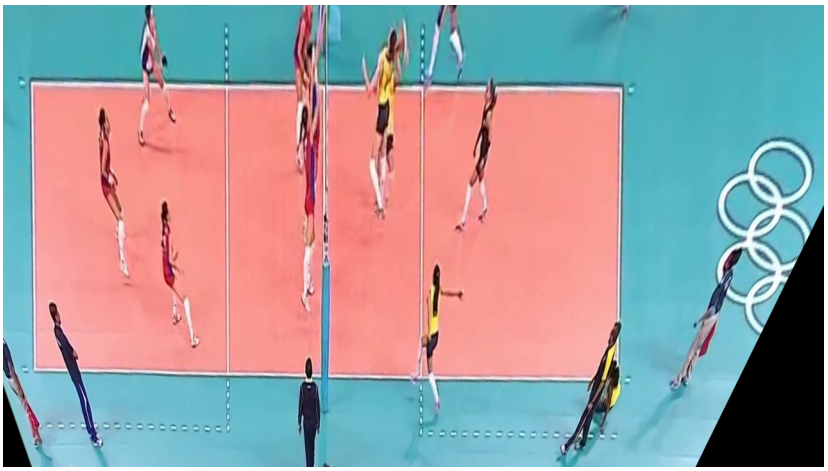

Figure 4.10: The projected image using Homography

```
self.court_homography = cv2.perspectiveTransform(pos_players , H)[0]
```

pos_players is an array containing the feet position of the players, they are points, estimated by using the middle point of bottom borders of their bounding boxes.

In some scenes there are some occlusions that made the detection of a player very difficult, especially in blocking actions where players are very close and make the same movements, overlaying each other in symmetric manner.

To cope with this problem, we lower the threshold of the confidence in person detection in first steps. It was not a big problems in training (because we have the initial bounding boxes where the players are), but in the prediction this can be a problems.

We exploit our detector in it, since now we have the position of each player in the court we can distinguish the player that are in the left or in the right side very easily. Moreover, thanks to the detector we can calculate more precisely also the distance from the borders and the net. This is very important, because some other people are very close to the court (coaches and referees) and they can be wrongly recognized as players.

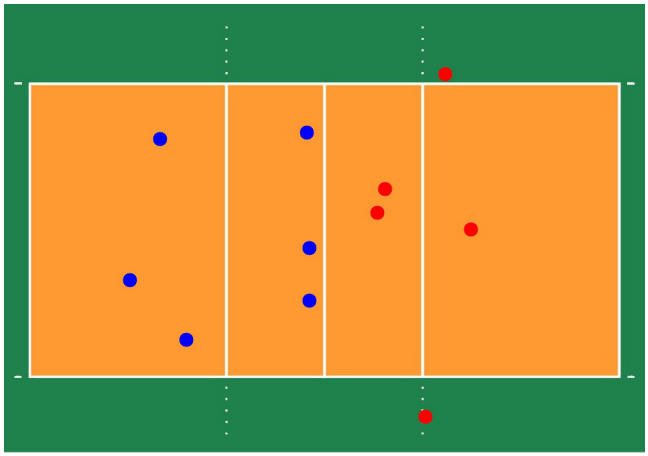

Figure 4.11: The keypoint positions using Homography

To get rid of false positives in the court, we adopted some simple rules, assigning scores according to the score in the detection task or to the distance from the net and the distance from the sidelines.

If a person is detected over the court, we can't have doubt: that's a player.

Problem is when there are detection outside. We can't exclude that those are players, because sometimes players go out form the court for different reasons: to retrieve a ball in a defense action, but very often there take a run to jump before spiking.

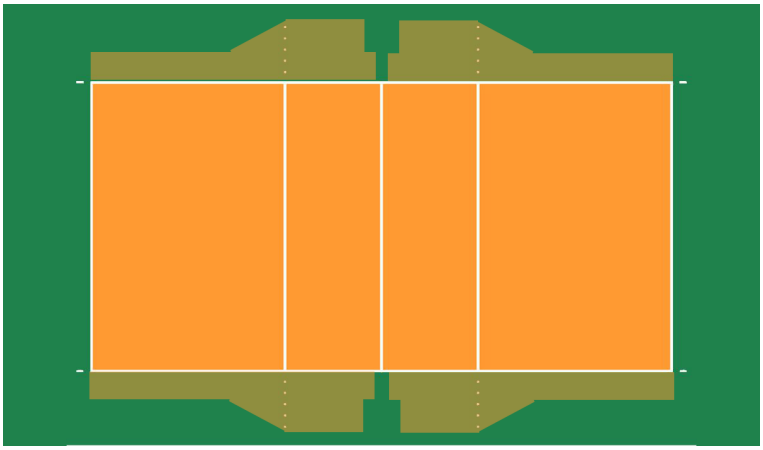

Figure 4.12: In orange, the extension of the court according the probability of the player presence

More a detected person is far away from the middle of the court (where there is the net), less probability, that is a player.

We augmented in some way the court area along the sidelines where we expect there can be players (Fig. 4.12)

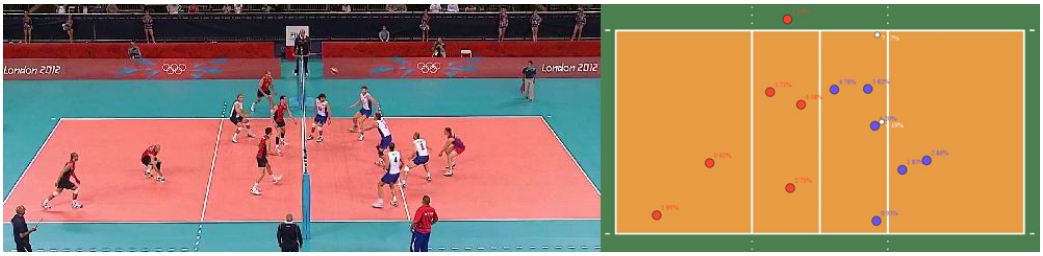

Figure 4.13: Position assignment according to the side and probability

The scores are used also after in an ordered list grouped by court side: only the first six players for side are drawn in the 2D model. Otherwise it could be that in a more crowded side of the court, there were more players with higher score than the other side.

4.9 Labeling model

Now we have for each sequence and for each frame in it a lot of data along the time: the player's position in 2D projected image, the poses of the players, the labels for each of them about the action they are performing.

There are networks that would let to detect action type in sports, also Gluon has got a pre-trained model ready to use for this task [8].

But we wanted to explore the possibility to analyze and recognize each action type starting from coordinates of all the poses along the time that we can collect by the alpha pose detector.

In fact, for each sequence we had all the poses of each player along 41 frames. Our idea is that a action type can be recognizable also from a single frame, but having the sequence also along the time of all the poses, we could exploit also the temporal dimension. Having a detector working on numerical data and not on frames in a video sequence has less data to analyze and it's faster.

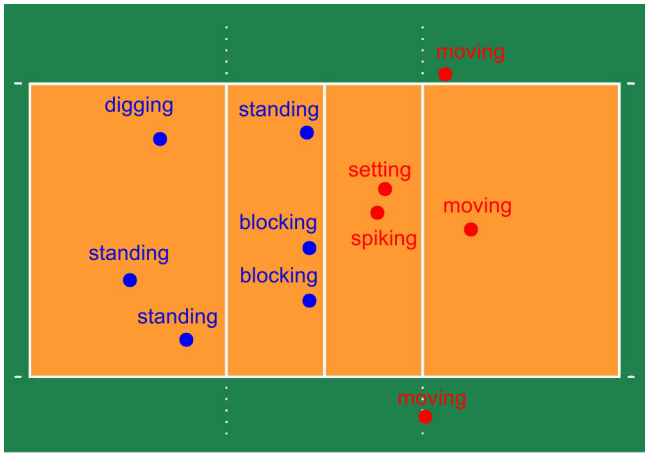

Figure 4.14: Our goal: having a bird view of players and their actions

For each player we had 41 poses, each pose has 17 (x,y) coordinates of skeleton joints. So, we had action information in two dimensions and wanted use it.

4.9.1 Pose encoding

The encoding of data is crucial, because we can make a representation of the reality in different way. In machine learning is very important how to encode data and to keep a semantic correspondence between the real world and your numeric encoding.

We tried different type of encoding but mainly two type: one with the coordinates of points, the other one with the cosine of the angles between the joints.

Encoding with Coordinates

It's the easier way because you can put in the network the data that you have already collected, hoping it understand the relative position of each player.

We tried to do in different manners to test all the possibilities and if there were some difference in using a method rather than another.

The first one was to get the poses coordinates as they were: with absolute values over the images.

The second method was to find the center of the each bounding box and calculating the relative coordinates from it, normalizing the values between 0 and 1 all the values.

This method has some problems, because the center of the bounding boxes along the time is not always the same, not only because the area occupy by a person can change according his body position, but also because the detector could get wrong the real dimension of the area (also per occlusion problems)

Comparing the pose coordinates frame by frame, with a center that can change continuously respect to real center is not so stable.

A solution that we tried has been instead of the center of the bounding box, to use a joint of the player, and specifically a joint that is in a center position for the body (ankle seems a good candidate).

Another help in encoding could be to treat all the players with the same point of reference respect to court side.

The players on the left side make the same kind of actions of the players on the right side, but flipped with respect to the net. It is like they are in front of a mirror.

To simplify the task, we flipped all the data coming from a side: we could do it because we knew which players where on the left and which on the right, thanks our court detector and the homography.

Another reduction of data dimension for the input of the network was to exclude some joints: for example quite often the gesture in volleyball are symmetric, digging, setting, blocking. Only spiking is not, but the most relevant variations are related to the arms (or better: the movements of the arms are enough to understand that that player is performing a spike).

So we kept in consideration to analyze only about 10 over 17 keypoints positions, excluding one side of the human body.

Encoding with Angles

Angle probably is a better encoding than the coordinates one.

Coordinates encoding is very heavy compared to the angle one: each point has to be encoded with 2 values (x,y). It means that we need 34 numbers to

define 17 joints. For angles (or cosine of angle) we need only 17.

Moreover angle representation is space invariant, scale invariant and rotation invariant (also if here maybe it doesn't matter).

Nevertheless it's a good definition about what we need: all the action types provide specific pose, that are quite standard. Measuring the width of the corners on specific pose should be enough to determinate a schema for all the different action types.

Again as vertex also for this encoding we used a ankle joint as reference. We we encoded all the joints vertex with their connections, we tried also to reduce the number of the encoding (keeping only the angle of legs and arms for example) or adding new correlations between body joints not naturally connected, but that could be very informative (for example the wrist with the knee).

In some test we added also some distance (for example wrist from foot) and also a movement factor, using as input the variation of her position respect to the court: in fact, if we encode every player using an internal point of reference, we lost movements information of that player in the context of court.

4.9.2 Results

We tried to use different type of simple model, with not so much hidden layers: CNN with 1D-ConvLayer, LSTM and Dense networks.

Related work use also Graph Convolutional networks or Spatio Temporal Graph Convolutional Networks for gesture detection.

In all our tests, we didn't get have a good prediction output.

We changed a lot of data encoding or detector models and then we investigated better over the dataset that we made up. In the end we found that the detection task worked quite well, and also the tracking task.

But the pose detection didn't work so well: we found that for a lot of player, pose detection didn't work (no output).

Some problems were due to rapid movements of the players (especially in spiking), this didn't let the detector to recognize the poses.

In other cases, the closeness between teammates confused the detector

Another problems was that the labeling on the original dataset had different logic of naming, that overlap each other, for example we can have a kind of movements that in dataset is labeled as "moving" but also as "spiking". One label didn't exclude the other. This can be quite ambiguous for the detector in training, therefore we tried to exclude all the action types out of Spiking, Digging and Blocking, during the training but results didn't

change: our dataset was not so informative to complete the action detection task.

Chapter 5

Conclusion

Starting from a problem of action type detection in volleyball, we have tested different techniques in machine learning and computer vision applied to sports.

The sub tasks are the detection of the ball, the players, and the playing court and the player's tracking in the video.

Our goal was to be able to collect as much information as possible starting from a simple video recordings with one camera on the long side of the court.

We leveraged a public dataset with several sequences of 50 different international volleyball matches. Each sequence had the middle frame labeled with action type over all the players, but all the other frames were unlabeled .

We augmented the information over this dataset, adding players tracking along the time on the other frames, and used the frames of the dataset to also label the court area.

We performed detection and pose tasks with Gluon CV toolkit using pretrained models and we created a custom centernet network from scratch to perform the court detection task.

For the goal of the project and also for the intermediate steps, recognizing precisely the shape of the court of the game was quite crucial.

For this reason, we created a dataset, labelling it with different images containing volleyball match scenes from the previous dataset, annotating 10 keypoints delimiting the playing court (4 corners of the rectangle, the middle line and the two lines that divide each side to the "3 meters" from the net).

Related works in sports court detection task, often use algorithms from classic computer vision field (Canny with Hough), which are very sensitive to environment variations like brightness or to the changes in camera shot (zoom or moving) and in the most of case they have to be calibrate in site.

Instead, our centernet model by a data-driven approach, it's able to rec-

ognize the 10 keypoints of the joints in a volleyball court and performs the task very well.

It's very robust to variations of points of view and perspective. It can find also ghost points out the frame (predicting where they are in the space not in frame) Thank to the precision of our detector we are able to calculate the homography of the frame on a 2D court model in bird view and to draw over it all the players.

We tried also to train a network to recognize action type in volleyball starting from coordinates that define players's poses along the time. We didn't succeed in it, mainly because the quality of the dataset that we have built is limited, due to the complexity to retrieve poses from images so crowd and using only one camera.

Chapter 6

Appendix

6.1 The volleyball dataset

Dataset url

https://github.com/mostafa-saad/deep-activity-rec

File readme.txt

We collected a new dataset using publicly available YouTube volleyball videos. We annotated 4830 frames that were handpicked from 55 videos with 9 player action labels and 8 team activity labels. We used frames from 2/3 rd of the videos for training, and the remaining 1/3 rd for testing. The list of action and activity labels and related statistics are tabulated in following tables.

Group Activity Class	No. of Instances
Right set	644
Right spike	623
Right pass	801
Right winpoint	295
Left winpoint	367
Left pass	826
Left spike	642
Left set	633

Action Classes	No. of Instances
Waiting	3601
Setting	1332
Digging	2333
Falling	1241
Spiking	1216
Blocking	2458
Jumping	341
Moving	5121
Standing	38696

Notes

The dataset is 55 videos.
Each video has a directory for it with sequential IDs (0, 1...54)

Train Videos: 1 3 6 7 10 13 15 16 18 22 23 31 32 36 38 39 40 41 42 48 50 52 53 54
Validation Videos: 0 2 8 12 17 19 24 26 27 28 30 33 46 49 51
Test Videos: 4 5 9 11 14 20 21 25 29 34 35 37 43 44 45 47

- Inside each video directory, a set of directories corresponds to annotated frames (e.g. volleyball/39/29885)
- Each frame directory has 41 images (20 images before target frame, target frame, 20 frames after target frame)
- You can use such window as your temporal window.
- Scenes change in fast manner in volleyball, hence frames beyond that window shouldn't represent belong to target frame most of time.
- In our work, we used 5 before and 4 after frames.
- Each video directory has annotations.txt file that contains selected frames annotations.
- Each annotation line in format: Frame ID Frame Activity Class Player Annotation Player Annotation ...
- Player Annotation corresponds to a tight bounding box surrounds each player
- Each Player Annotation in format: Action Class X Y W H

Informal info

Following is a list of youtube videos. Downloading the video and extracting the frames with correct resolution, bit rate..etc might be challenging.

The frame sequences in each folder come from this matches:

Folder	Match	Competition
0	China vs USA	FIVB World Grand Prix 2015 720P
1	Brazil vs Russian Fed - Women's Volleyball Quarterfinal	London 2012
2	Brazil vs USA - Women's Volleyball Final	London 2012 Olympic Games
3	Live - Brazil vs China	FIVB World Grand Prix Finals 2015
4	Live - Brazil vs France	FIVB Volleyball World League Finals 2015
5	Live - Brazil vs Japan	FIVB World Grand Prix Finals 2015
6	Live - Brazil vs Russia	FIVB World Grand Prix Finals 2015
7	Live - Brazil vs USA	FIVB Volleyball World League Final 2015
8	Live - China vs Italy	FIVB World Grand Prix Finals 2015
9	Live - China vs Russia	FIVB World Grand Prix Finals 2015
10	Live - France vs Poland	FIVB Volleyball World League Final 2015
11	Live - Italy vs Brazil	FIVB World Grand Prix Finals 2015
12	Live - Italy v Turkey	FIVB Volleyball Girls' U18 World Championship Peru 2015
13	Live - Italy vs Japan	FIVB World Grand Prix Finals 2015
14	Live - Italy vs Russia	FIVB World Grand Prix Finals 2015
15	Live - Japan vs China	FIVB World Grand Prix Finals 2015
16	Live - Japan vs Russia	FIVB World Grand Prix 2015
17	Live - Poland vs Italy	FIVB Volleyball World League Final 2015
18	Live - Serbia vs France	FIVB Volleyball World League Finals 2015
19	Live - Serbia vs Poland	FIVB Volleyball World League Final 2015
20	Live - USA vs Brazil	FIVB World Grand Prix Finals 2015
21	USA vs China 26 July 2015 - Week 5 Final Round - 2015	FIVB World Grand Prix
22	Live - USA vs France	FIVB Volleyball World League Final 2015
23	Live - USA vs Italy	FIVB World Grand Prix Finals 2015
24	Live - USA vs Japan	FIVB World Grand Prix Finals 2015
25	Live - USA vs Poland	FIVB Volleyball World League Finals 2015
26	Live - USA vs Russia	FIVB World Grand Prix Finals 2015
27	Live - USA vs Serbia	FIVB Volleyball World League Final 2015
28	Men's Volleyball Pool B - BRA v SRB	London 2012 Olympics
29	Men's Volleyball Pool B - BRA v USA	London 2012 Olympics
30	Men's Volleyball Preliminary - BRA v GER	London 2012 Olympics
31	Men's Volleyball Preliminary Round - AUS v POL	London 2012 Olympics
32	Men's Volleyball Preliminary Round - RUS v SRB	London 2012 Olympics
33	Men's Volleyball Preliminary Round USA v TUN - Highlights	London 2012 Olympics
34	Men's Volleyball Quarter-Finals - BUL v GER	London 2012 Olympics
35	Men's Volleyball Quarterfinals - ITA vs USA	London 2012 Olympics
36	Men's Volleyball Quarter-Finals - POL v RUS	London 2012 Olympics
37	Volleyball - Men - BRA-RUS	London 2012 Olympic Games
38	Volleyball - Men - BRA-TUN	London 2012 Olympic Games
39	Volleyball - Men Bronze Final	London 2012 Olympic Games
40	Volleyball - Men Gold Final - VC RUS-BRA	London 2012 Olympic Games
41	Volleyball - Men RUS-USA	London 2012 Olympic Games
42	Volleyball - Men SF 2 BRA-ITA	London 2012 Olympic Games
43	Volleyball Men's Preliminary Pool A Italy v Bulgaria - Full Replay	London 2012 Olympics
44	Volleyball - Women Bronze Final JPN-KOR	London 2012 Olympic Games
45	Volleyball - Women SF 2 BRA-JPN	London 2012 Olympic Games
46	Women's Volleyball Pool A - ITA v JPN	London 2012 Olympics
47	Women's Volleyball Pool A - ITA v RUS	London 2012 Olympics
48	Women's Volleyball Pool A - JPN V GBR	London 2012 Olympics
49	Women's Volleyball Pool B - BRA v TUR	London 2012 Olympics
50	Women's Volleyball Pool B - Korea v USA	London 2012 Olympics
51	Women's Volleyball Pool B - CHN v KOR	London 2012 Olympics
52	Women's Volleyball Pool B - USA v Brazil	London 2012 Olympics
53	Women's Volleyball Preliminary Round - ITA v ALG	London 2012 Olympics
54	Women's Volleyball Quarter Finals - JPN v CHN	London 2012 Olympics

6.2 Basic Volleyball Rules for Playing the Game [6]

- A court of game divided by a net in the middle
- Six players on the court for each team, that stay on their half of the court
- Each team can hit the ball for a maximum of three times per side and then the ball has to go on the opponent's side of the court.
- A player may not hit the ball twice in succession (A block is not considered a hit)
- A ball hitting the floor inside or the boundary line is "in", is "out" if touches everything else.
- It is illegal to catch and hold the ball, the touch has to be quick
- At higher competition, the officiating crew may be made up of two refs, line judges, scorer, and an assistant scorer

Players can perform six basic skills: serve, pass, set, attack, block and dig. Each of these skills comprises a number of specific pose:

Serve

A player stands behind the inline and serves the ball, in an attempt to drive it into the opponent's court.

Pass

A player making a forearm pass, it also called reception, it includes not only preventing the ball from touching the court but also making it reach the position where the setter is standing quickly and precisely.

The pass involves two specific techniques: underarm pass (or bump), where the ball touches the inside part of the joined forearms and overhand pass, where it is handled with the fingertips, like a set, above the head.

Set

The set is usually the second contact that a team makes with the ball. The main goal of setting is to put the ball in the air in such a way that it can be driven by an attack into the opponent's court. The setter coordinates the offensive movements of a team, and is the player who ultimately decides which player will actually attack the ball.

As with passing, one may distinguish between an overhand and a bump set. As with a set or an overhand pass, the setter/passer must be careful to touch the ball with both hands at the same time. If one hand is noticeably late to touch the ball this could result in a less effective set, as well as the referee calling a 'double hit' and giving the point to the opposing team.

Attack

The attack, also known as the spike, is usually the third contact a team makes with the ball The object of attacking is to handle the ball so that it lands on the opponent's court and cannot be defended. A player makes a series of steps (the "approach"), jumps, and swings at the ball.

A 'bounce' is a slang term for a very hard/loud spike that follows an almost straight trajectory steeply downward into the opponent's court and bounces very high into the air. A "kill" is the slang term for an attack that is not returned by the other team thus resulting in a point.

Block

Blocking refers to the actions taken by players standing at the net to stop or alter an opponent's attack. A block that is aimed at completely stopping an attack, thus making the ball remain in the opponent's court, jumping and reaching to penetrate with one's arms and hands over the net and into the opponent's area.

Dig

Digging is the ability to prevent the ball from touching one's court after a spike or attack, particularly a ball that is nearly touching the ground. In many aspects, this skill is similar to passing, or bumping

Bibliography

[1] URL: https://www.fivb.com/en/about/news/hosts-thailand-stun-world-champions-china-in?id=50097. (accessed: 11.11.2021).

[2] *Centernet in Mxnet*. URL: https://github.com/Guanghan/mxnet-centernet. (accessed: 11.11.2021).

[3] Julian Chuaby. *An Overview Of Smart Tennis Courts 2020*. URL: https://sportstechnologyblog.com/2020/09/07/an-overview-of-smart-tennis-courts-2020/. (accessed: 11.11.2021).

[4] *Computer Vision in Sport*. URL: https://www.sportperformanceanalysis.com/article/computer-vision-in-sport. (accessed: 11.11.2021).

[5] Intel Corporation. *Hough Line Transform*. URL: https://docs.opencv.org/%202.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html. (accessed: 11.11.2021).

[6] Joel Dearing. *Volleyball fundamentals. Champaign, IL: Human Kinetics*. 2003. ISBN: 0736045082.

[7] *Expert AI*. URL: https://www.expert.ai/blog/machine-learning-definition/. (accessed: 11.11.2021).

[8] *Getting Started with Pre-trained TSN Models on UCF101*. URL: https://cv.gluon.ai/build/examples_action_recognition/demo_tsn_ucf101.html#sphx-glr-build-examples-action-recognition-demo-tsn-ucf101-py. (accessed: 11.11.2021).

[9] *Gluon*. URL: https://gluon.mxnet.io. (accessed: 11.11.2021).

[10] *GluonCV*. URL: https://cv.gluon.ai. (accessed: 11.11.2021).

[11] *Hawk-eye innovations. Hawk-eye innovations tennis*. URL: https://www.hawkeyeinnovations. (accessed: 11.11.2021).

[12] Mostafa S. Ibrahim et al. "A Hierarchical Deep Temporal Model for Group Activity Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[13] Mostafa S. Ibrahim et al. “Hierarchical Deep Temporal Models for Group Activity Recognition.” In: 2016.

[14] Colby T. Jeffries. *Sports Analytics With Computer Vision*. 2018. URL: https://openworks.wooster.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=10456&context=independentstudy. (accessed: 11.11.2021).

[15] *Keypoint Evaluation*. URL: https://cocodataset.org/#keypoints-eval. (accessed: 11.11.2021).

[16] Michael Lewis. *Moneyball: The Art of Winning an Unfair Game*. 2003.

[17] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science, 1997.

[18] *Moneyball*. URL: https://en.wikipedia.org/wiki/Moneyball. (accessed: 11.11.2021).

[19] *Posenet - Github*. URL: https://github.com/tensorflow/tfjs-models/tree/master/posenet. (accessed: 11.11.2021).

[20] Kumba Sennaar. *Artificial Intelligence in Sports – Current and Future Applications*. 2019. URL: https://emerj.com/ai-sector-overviews/artificial-intelligence-in-sports/. (accessed: 11.11.2021).

[21] *SportVu*. URL: https://www.statsperform.com/team-performance/basketball/. (accessed: 11.11.2021).

[22] Steven Wu and Luke Bornn. “Modeling offensive player movement in professional basketball”. In: (2017).

[23] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. “Objects as Points”. In: *arXiv preprint arXiv:1904.07850*. 2019.

Acknowledgement

First of all, I would like to thank Luca, for his willingness to help everyone, me included.

Luca Lorello, Nicolaas Ruberg, Salman Razzaq were three important travel buddies of the last two years at Unibo, despite the covid we manage to meet, study together and have our coffee breaks.

Figure 6.1: An example in multipose detection with four A.I. students visiting the Certosa in Bologna

I would also like to thank Fabrizio Rossini and Massimo Righi of the Lega Pallavolo Serie A (the Italian Volleyball League) for the availability of resources and data over the last many years.

Last but not least, Tommaso and Valentina.