1 2 9 0

UNIVERSIDADE D
COIMBRA

João Francisco Fernandes Pereira Ruivo

Volleyball Action and Activity Recognition

Dissertação no âmbito do Ramo de Computadores do Mestrado de Engenharia Eletrotécnica e de Computadores orientada pelo Professor Luís Alberto da Silva Cruz, coorientada pelo Dr. Matheus Vicari (iSPORTiSTiCS) e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2023

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Volleyball Action and Activity Recognition

João Francisco Fernandes Pereira Ruivo

Coimbra, September 2023

Volleyball Action and Activity Recognition

Supervisor:

Luís Alberto da Silva Cruz

Co-supervisor:

Matheus Vicari

Jury:

Nuno Miguel Mendonça da Silva Gonçalves

Luís Alberto da Silva Cruz

Jorge Manuel Moreira de Campos Pereira Batista

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2023

Acknowledgements

Firstly, I'm obliged to thank my family: my mother who is ever loving and supportive, my father who works hard and motivates me to work just as hard, my two brothers, the two smartest people I know, who always have great advice to give. You will surely be very successful in your academic and professional lives.

I can't speak of family without including all my dear friends, truly a second family that is sure to cheer me up when things aren't going smoothly. I'm grateful to have so many great memories with each of your unique selves.

I also thank the people at iSPORTiSTiCS, namely Vini and Matheus, for trusting me with this research work. I appreciate the opportunity, and it was a pleasure to collaborate in this dissertation.

Lastly, I thank my supervisor Dr. Luís Cruz for welcoming me and my ideas and for helping me through my doubts and uncertainties through this dissertation.

Abstract

Deep Learning (DL) has been used for sports video understanding, and the developed models for group activity recognition that can be used in volleyball scenarios are constantly improving with new proposals. With all the already available models, we select the ones that are accurate but also easy to deploy in a real scenario, and analyze and test them in the benchmark dataset Volleyball Dataset (VD), as well as in new data, with annotations obtained with DL in an unsupervised manner. These annotations are important to have a truly automatic method for video analysis, thus, we also study and test the best ways to detect the players and the ball in a volleyball scenario, track them over the time frame and detect their keypoints (or joints). These three previous tasks generate necessary data for the activity and action classification models. We then feed the detections to the classification models in order to classify unannotated videos, which is the goal of any application that uses machine learning for sports analysis and understanding. From all the tested models, the keypoint modality is the most successful with an accuracy of 95.00% in the VD, but with an accuracy of 63.33% while running inference on unsupervisedly annotated data. We observe that object detections can be challenging for the volleyball scenario, with player detection achieving an F1-Score of 75%. In conclusion, action and activity classification on unannotated videos can be performed, although not with the same accuracy that is observed and reported in benchmark conditions, where the players' bounding boxes are manually annotated.

Keywords: Group Activity Recognition, Human Action Recognition, Object Detection, Object Tracking, Keypoint Detection

Resumo

A Aprendizagem Profunda (DL) tem sido recorrentemente utilizada na análise de vídeos desportivos, e os modelos desenvolvidos para o reconhecimento de atividades em grupo que podem ser usados em cenários de voleibol estão constantemente a melhorar, com novas propostas. De todos os modelos já disponíveis, selecionamos aqueles que têm maior precisão, tendo também em conta a facilidade de implementação num cenário real, analisando-os e testando-os no dataset de referência Volleyball Dataset (VD), bem como em novos dados, com anotações obtidas através de DL de forma não supervisionada. Estas anotações são importantes para haver um método verdadeiramente automático para a análise de vídeos. Por isso, também estudamos e testamos as melhores formas de detetar os jogadores e a bola num cenário de voleibol, acompanhá-los ao longo do tempo (tracking) e detetar os seus keypoints (ou articulações). Estas três tarefas geram dados necessários para os modelos de classificação de atividades e ações. Em seguida, usamos as deteções obtidas nos modelos de classificação para classificar vídeos não anotados, que será o objetivo de qualquer aplicação que utilize métodos automáticos para análise e compreensão de situações em vídeo de desporto. De todos os modelos testados, a modalidade de keypoints é a mais bem-sucedida, com uma accuracy de 95,00% no VD, mas com uma accuracy de 63,33% quando aplicado em dados anotados de forma não supervisionada. Observamos que as deteções de objetos podem ser desafiantes para o cenário de voleibol, com a deteção de jogadores a atingir um F1-Score de 75%. Em conclusão, a classificação de ações e atividades em vídeos não anotados pode ser realizada, embora não com a mesma precisão observada e relatada em condições de referência, onde as bounding boxes dos jogadores são anotadas manualmente.

Palavras chave: Reconhecimento de Actividade em Grupo, Reconhecimento de Ações Humanas, Deteção de Objetos, Seguimento de Objetos, Deteção de Keypoints

"He who asks is a fool for five minutes, but he who does not ask remains a fool forever."

— Chinese proverb

Contents

List of Acronyms

BS Batch Size

CNN Convolutional Neural Network

COCO Common Objects in Context

CSTT Clustered Spatial-Temporal Transformer

DIN Dynamic Inference Network

DL Deep Learning

DR Dynamic Relation

DW Dynamic Walk

FIVB Fédération Internationale de Volleyball

FN False Negative

FP False Positive

GAR Group Activity Recognition

GNN Graph Neural Network

HAR Human Action Recognition

LR Learning Rate

MHAP Multi-Head Attention Pooling

MLP Multilayer Perceptron

MOT Multiple Object Tracking

MOTA Multiple Object Tracking Accuracy

NLP	Natural Language Processing
OD	Object Detection
OKS	Object Keypoint Similarity
ROI	Region of Interest
SOTA	State-Of-The-Art
TP	True Positive
YOLO	You Only Look Once

List of Figures

List of Tables

1 Introduction

1.1 Context and Motivation

In sports analysis, a person must be performing an individual action during each collective activity. This collective activity recognition can be used to generate play-by-play analysis and highlights for broadcasting and media creation purposes, and can even be used for scouting, tactical evaluation and opposition analysis on behalf of a team's staff. However, the individual action recognition task has proven to be quite difficult, since most of the actions are very similar to each other, and on top of that, players are often occluded or heavily blurred due to movement. Fortunately, it's usually the group activity recognition that can be used to automatically extract information from a volleyball clip or frame.

The presented work was suggested by iSPORTiSTiCS, a company whose services consist on using artificial intelligence to automatize content creation for sports broadcasters, with the above described media content creation methods in mind. Among their current solutions, available at https://isportistics.com, tactical analysis and video clipping of highlights are those that take advantage of action and activity recognition in sports.

HAR and GAR are tasks accomplished by machine learning models. Just like typical image classification tasks, deep learning (DL) techniques are used to achieve the best results by training in large datasets. In order to accomplish these machine learning driven enhancements it is not only necessary to accurately classify the group activity and person action, but a pipeline to extract all the data for the model itself beforehand might be needed as well.

The GAR and HAR tasks in sports have been extensively studied and refined as DL technologies evolve[1]. However, there aren't studies available that explore how effective these models would be in a real application and how practical their inference in non-annotated videos would be; after all, if we want to use AI to automatize a process, we need to know how good the solution is at the actual problem, outside the controlled training environment.

1.2 Objectives

DL is currently emerging as the new way to automatize everything from content creation, language processing and image or video detection and classification, therefore, there are already many studies for action recognition, group activity recognition and person detection that report splendid results in benchmark datasets. These state of the art (SOTA) techniques were created through highly focused research, which implies an investment of time and resources towards obtaining efficient tools for the task in hand, and we aim to test some of them in a realistic end-to-end environment, where every step of the video processing is automatized. With this in mind, the objectives of the dissertation can be summarized as:

- Studying and testing SOTA deep learning models for GAR and HAR.
- Testing and selecting the best models for object detection, object tracking and keypoint detection.
- Developing an end-to-end architecture, where the input is a few frames or a short video clip of volleyball activity representing a group activity, that can classify said activity and each player's individual action.
- Fine tuning the existing models for HAR and GAR in order to achieve the most accurate results possible in real data with the tested tools.

This dissertation not only tests the efficiency of several deep learning models for GAR and HAR, but also takes into account the previous steps necessary to effectively run them. The best tools, measured in a few clips taken from videos that are not part of the training dataset, for detecting and collecting the necessary data for GAR/HAR are used to compile a means to classify actions from volleyball frames. So, besides studying the currently available models and their potential to be improved or further tuned, we also study multiple methods to detect and track the players and the ball in video frames and their effectiveness, as the final activity and action classification can only be as good as their actor's detections. Beyond player detection and tracking, some models[2][3] use keypoint modalities instead of only RGB features and because of this, we study the SOTA for person and object keypoint extraction too.

1.3 Dissertation Outline

Considering the main objectives described in Section 1.2, this Dissertation is organized as follows:

- **Chapter 1** (this chapter) presents the context and objectives for this Dissertation.

- **Chapter 2** lays the foundation of concepts that promote a good understanding of the proposed and addressed methodologies.
- **Chapter 3** reviews the DL GAR and HAR in volleyball, describing the SOTA solutions for some collective activity models.
- **Chapter 4** presents the proposal of an end-to-end architecture that takes as input a few video frames and outputs the multiple activities that were detected, as well as the tests conducted with object detectors and trackers, keypoint detections and GAR and HAR models.
- **Chapter 5** summarizes the conclusions that can be inferred from the tests conducted and provides insight into possible future work.
- **Annex A** shows multiple results with different runs of YOLO object detectors.
- **Annex B** shows multiple studies of the COMPOSER model, where the input of the model is slightly changed.

2 Background Information

The current DL techniques are increasingly complex in all fields. DL is a hot-topic where a lot of research focuses on, and the models that are currently achieving SOTA results in the GAR, HAR and all detection tasks are also very complex; with some of them using specific concepts that can be rather abstract. In this section, we intend to lay a foundation for a better understanding of all the key concepts that were introduced by the authors of the models we use ahead. We start by addressing simple DL related concepts and work our way through each of the more complex ones.

However, we first address how volleyball games are broadcast, and the implications that this has on the way we should expect the model's input data to be formatted.

Volleyball games are usually recorded from a side view of the field, as in figure 2.1, which is actually very convenient for the training of ML models. By not having many different angles that we have to worry about and that the model would have to generalize to, it's fairly easier to train a robust model on a reasonable sized dataset. In fact, the Volleyball Dataset (VD)[4] has been used thoroughly for group activity classification for a long time[1] and contains exactly frames like this.

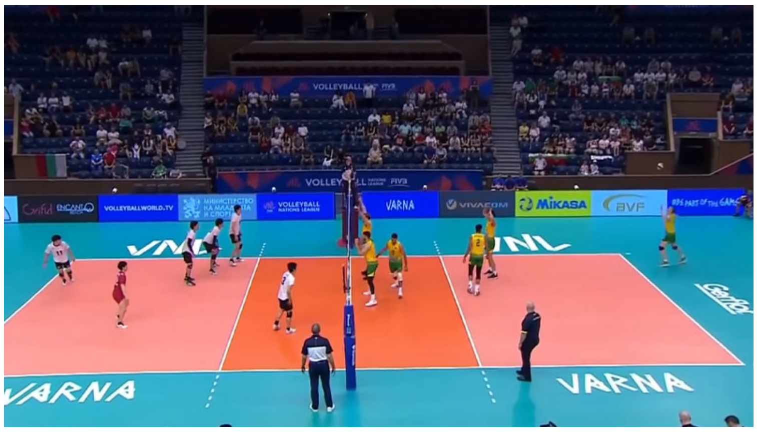


Figure 2.1: Example of a frame from a volleyball clip as expected to be seen in streaming or broadcast services.

2.1 Volleyball Dataset

For the training and testing of the developed architectures and their tuning, the openly available Volleyball Dataset, introduced by Ibrahim et al.[4], is used. The dataset is adequate to the problem, as it contains 4830 frames from 55 online available videos, which amounts 59GB, containing mostly volleyball games from the 2012 London Olympics and the Fédération Internationale de Volleyball (FIVB) World Grand Prix Finals 2015. These are mostly frames with resolution of 1280x720, although a few are 1920x1080, that have been annotated with 8 team activity labels: *right set, right spike, right pass, right winpoint, left winpoint, left pass, left spike, left set*; and 9 player action labels: *waiting, setting, digging, falling, spiking, blocking, jumping, moving, standing.* The dataset is not balanced, as some classes have a lot more instances than others, which happens both for group activity classes, as shown in table 2.1, and action classes, as shown in table 2.2.

Of the annotated activities, *pass*, or also referred to colloquially as reception, denotes a player controlling the incoming attack (serve or spike) and directing the ball to their teammates; pass should be associated with the digging action. Following a pass, the player that receives the ball usually *sets* it, by pushing it into the air, ideally positioning the ball for the next teammate to perform an attack, also known as *spike*, which in turn depicts a player jumping and hitting the ball with force over the net into the opponent's court. Finally, *winning a point* happens when the ball lands on the opposite court without being defended, or in rarer cases, when the opponent makes an error like hitting the ball out of bounds.

Table 2.1: Number of instances of each group activity class. The dataset assumes that the video is only recorded after the service and stops recording after the winpoint, as the dataset contains only the actions during the match point up until one team wins the point.

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

Table 2.2: Number of instances of each action class.

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

Demirel and Ozkan[5] report that the VD contains a few incorrect annotations, 497 in total, and provide the corrected annotations. Because our goal is to have a practical application for GAR and HAR it's important that we test on correctly annotated data to actually know how good the models can be in test and inference conditions. With their reannotations, the number of instances is changed to the following:

Table 2.3: Number of instances of each group activity class after correcting the annotations.

Group Activity Class	No. of Instances
Right set	596
Right spike	640
Right pass	830
Right winpoint	297
Left winpoint	368
Left pass	831
Left spike	654
Left set	605

In the dataset in hand, each frame directory contains the 20 video frames before the target frame, also referred to as the central frame, (an example of which can be seen in figure 2.2), as well as the 20 video frames after. The annotations file contains a Frame ID, the label and player bounding boxes in the format of $[x_{center}, y_{center}, width, height]$. In our training we can use all of these 40 frames, but we can also use fewer.

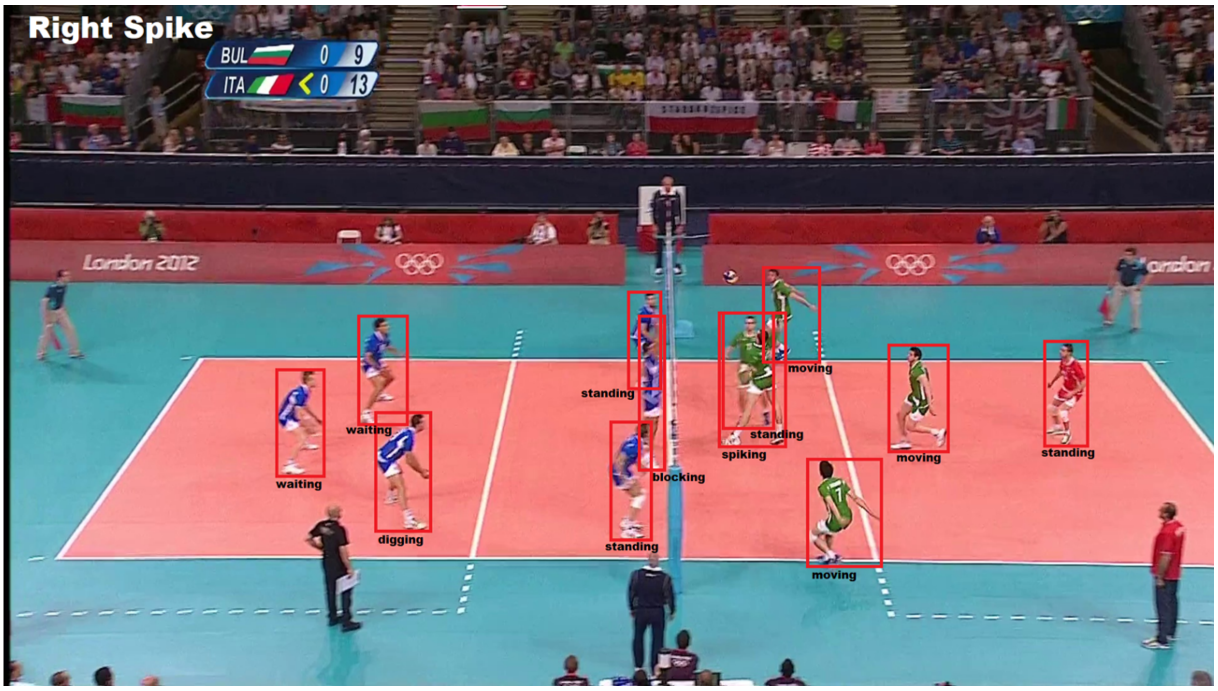


Figure 2.2: Annotated target frame example.

The VD is usually split in train and test:

- 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 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

And all the models used were also implemented accordingly with this split.

2.1.1 Ball Annotations

The ball's position in a frame and its trajectory along the time dimension can also be used as important features for both GAR and HAR. Perez *et al.* have manually annotated the ball's position in each frame of the Volleyball Dataset[6]. It should be noted that annotating the ball's position can be done manually, but detecting it with machine learning (ML) and computer vision techniques can be an exceptionally difficult task, as addressed later in chapter 4, because at times the ball travels at high speeds, which leads to it being heavily blurred or even totally missing from the frame; not to mention the possibility of the ball leaving the camera's field of vision completely during a pass or set.

2.1.2 Player Tracking Annotations

Since video action recognition must take into account the temporal dimension, it's important to correctly identify each object that appears sequentially in the frames. These annotations are also made available in [7]. Unlike the ball detection, player detection and tracking aren't as challenging, and can be done with existing tools.

Thanks to these annotations we can also train and evaluate the detecting and tracking tools on the VD and therefore know how accurate these detections are in the context of volleyball video.

Although, as mentioned before, the VD contains 40 frames for each activity, the tracking annotations are only for 20 frames: 9 before and 10 after the target frame. This means that the maximum number of frames we can use for training is 20, since we don't have tracking annotations for the remaining 20.

2.1.3 Data Formats

There are mainly two modality approaches to the GAR and HAR problems: RGB features and human pose features, or keypoints. The former is easier to input to models, as that's just the raw data from the frames, usually accompanied by coordinates of each player. On the other hand, human pose estimation is a challenge by itself in general, but becomes extraordinarily difficult in volleyball video, as it's very common to have actors occluding each other or blurred actors. Nevertheless, keypoint modalities can yield great results with lighter models[2][3].

2.2 Deep Learning Fundamentals For Video Classification

Be it image classification, natural language processing (NLP) or video classification, deep learning is currently at the center of every ML model. Since one can intuitively think about video being but a sequence of images, it can be useful to first understand how DL is used for image classification, and that is commonly with convolutional neural networks (CNN).

Contrary to images, however, video processing also demands modeling of the temporal dependencies embedded in the sequence of frames that constitutes the video, and for that, three-dimensional CNNs (3D-CNNs) can be used in an architecture that harmonizes spatial and temporal features. 3D-CNNs apply convolutions to 3 dimensions instead of the usual 2: the third being the time dimension created by the sequence of frames, which allows the model to learn spatio-temporal features of the input.

Because both NLP and video classification can be seen as inputs with relevant temporally sequential features, it should also make sense that attention mechanisms, that have been proved extremely efficient for NLP, are also capable of great accuracy in image and video classification.[8]

In this section, we start by explaining the foundations of artificial neural networks and then move on to explaining the currently used mechanisms in video classification.

2.2.1 Linear Layers and Activation Function

Linear layers, also called dense or fully connected layers, apply linear transformations, a matrix multiplication with the weights and a sum of a bias, to their input. These linear layers are used sequentially, connecting the output of one layer to the input of the other one, in a way that the model learns abstract, complex and hierarchical representations of the input data along its layers.

Because we still want each layer to contribute to the learning process, if we only use linear layers alone, their whole sequence could be represented with just one linear transformation, defeating the purpose of having multiple linear layers. To introduce non-linearity to the model and thus have each layer contribute to the learning process, an activation layer is necessary after each linear transformation to add non-linearity. Typically, ReLU (equation 2.1), defined as the positive part of its argument, is the function used as the activation layer for artificial neural networks.

Each linear layer applies its transformation as seen in 2.2.

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.1}$$

$$\mathrm{y} = \mathrm{ReLU}(W \cdot x + b) \tag{2.2}$$

Where x is the input, W the weight matrix, b the bias vector and y the output.

Since the ReLU activation function can skew the values towards 0 if many values of the function are negative, a simple normalization layer can be useful to normalize the input tensor to have zero mean and standard deviation one.

2.2.2 Backpropagation and Loss Function

To evaluate how good the model's predictions are compared to the real labels of the input we must define a loss function that measures the error of the model's predictions. In other words, we want to minimize this loss function's value by updating the values of the weight matrix (and the biases) that are used to calculate each layer's outputs.

Backpropagation calculates the gradients of the loss function with respect to the weights and biases starting at the output layer, and progressing towards the first layer, updating those parameters with the calculated gradients and the chosen optimizer in the direction that minimizes the loss value.

2.2.3 Cross-Entropy Loss Function

For classification problems the loss function is usually the cross-entropy. It measures the dissimilarity between the predicted classes probabilities and the true labels and can be expressed as:

$$\text{CrossEntropyLoss}(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.3}$$

where y is the true label and $\hat{y}$ the predicted label.

The penalty being logarithmic means that predicted classes with a low probability will lead to a very high loss value, while predicted classes with a probability very close to one will lead to loss values close to zero. Cross-entropy is the function that will be minimized by the optimizer in classification problems.

2.3 Convolutional Neural Networks

CNNs comprise convolutional layers to extract feature maps from data and fully connected layers that receive as input those feature maps. They have been frequently used in images, as different networks keep achieving SOTA results for image classification.

A convolutional layer uses kernels with learnable weights to apply convolutions along the image, sliding over it according to the chosen stride. These operations result in feature maps that capture the local patterns of the original data and that are usually fed through pooling layers that reduce their spatial dimensions. For example, if we use max-pooling in a feature map extracted from a patch of an image, the result will be the pixel with the highest value, or if we use average-pooling in that same feature map, the result will be the average value of the pixels in the feature map. Max-pooling is the most common pooling operation, as it extracts the most salient features from the feature map.

Following the convolutional layers are the fully connected layers that take as input the feature maps extracted with the convolutional layers. Proceeding as described previously in this chapter, they lead to the final classification layer. In order to have accurate predictions, the CNN now iterates the described process to minimize the loss function through backpropagation, just like any other DL model.

2.3.1 VGG16[9]

The architecture of VGG16 (Visual Geometry Group 16) consists in a deep stack of 16 layers, including 13 convolutional layers and 3 fully connected layers. Each convolutional layer is relatively small, consisting of 3×3 filters with a fixed stride and padding. VGG16 achieved remarkable success in image classification tasks, particularly during the ImageNet Large Scale Visual Recognition

Challenge (ILSVRC) in 2014, where it showcased the potential of deep neural networks for image understanding. VGG16 has since been used as a backbone for various computer vision tasks, such as object detection, image segmentation, and even transfer learning due to its strong feature extraction capabilities. In our case, the RGB based models that we run experiments on use VGG16 as the backbone feature extractor.

2.4 3D-CNN

CNNs are great for spatial feature learning, and 3D-CNNs are the intuitive progression that allows CNNs to also learn spatiotemporal features, which is crucial information in videos and particularly in action recognition[10]. Instead of sliding 2D kernels over the input data like CNNs, 3D-CNNs use 3D kernels that apply convolutions across the three dimensions of the input, as shown in Figure 2.3, resulting in 3-dimensional feature maps that highlight spatiotemporal patterns in the input data.

Just like regular CNNs, 3D-CNNs also include pooling layers that reduce the dimensions of the input, in an attempt to highlight the most relevant features processed. Still similarly to 2D-CNNs, the convolutions are followed by the fully connected layers that aggregate the learned spatiotemporal features with the final goal of classification.

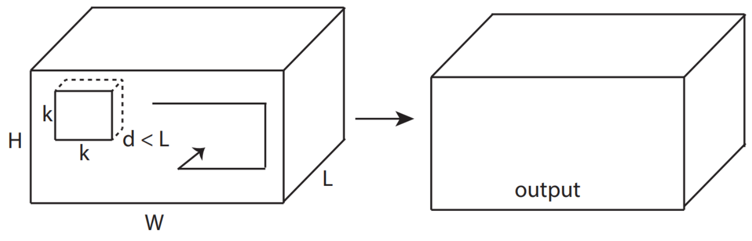


Figure 2.3: Visualization of a 3D-CNN with a kernel of size $k \times k \times d$ in a video with a resolution of $H \times W$ and L frames. *Source:* [11]

Training a 3D-CNN typically requires large datasets because they are very complex models that are prone to overfit to training data. Nevertheless, they were achieving SOTA results in a lot of action recognition in video datasets before transformer and attention based architectures started being more frequently adopted. 3D-CNNs still detain state of the art results in some GAR tasks[3].

2.5 Graph Neural Networks (GNN)

A graph is a mathematical representation consisting of nodes connected by edges where each node in the graph represents an entity or data point, and each edge represents a relationship between the nodes. In the context of GAR, usually GNNs come up with the Actor Relation Graph (ARG)[12], a way to model actors into graphs. After capturing every actor's features, ARG builds actor relation graphs, having each actor as a node and each edge as a scalar weight that depends on the two actor's visual features and their locations. Because there are many actors, there are also many relation graphs from a same set of actor features, in order to represent every interaction. Afterwards, a graph convolution is applied for the actual learning and inference purposes; a graph convolution can be seen as the previously mentioned CNN generalized to graph-structured data.

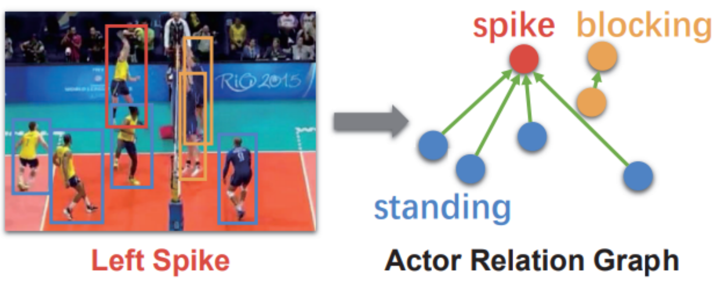


Figure 2.4: Visualization of a GNN in the context of a volleyball game. The nodes represent the players and the edges (green arrows) are the relations that exist between two actors.*Source:* [12]

2.6 Transformer

Attention mechanisms first started being used to model dependencies between the input data, something that should be very useful for action recognition. The pure-attention model introduced by Vaswani *et al.*, the transformer, is a model based on multi-headed self-attention, which has been very popular with NLP problems[13]. Self-attention computes a representation of the input sequence by relating its different positions and emphasizing the important parts of the data while despising irrelevant data.

Attention mechanisms involve three fundamental components: key, query and value. The keys are tokens that contain information about the input data and are used to calculate the attention weights. Meanwhile, the queries should contain information about context, in a way to have the keys better adjust themselves to attend to actually relevant features. The values can be seen as the tokens that the mechanism attends to, meaning that we calculate the attention weights to each value based on the keys and queries, outputting a weighted sum of the values.

The Transformer paper uses Scaled Dot-Product Attention to calculate attention taking the dot product between the queries and keys, scaling it by the dimension of the queries and keys and applying a softmax function to normalize the weights; which can be written as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

Where Q, K, and V represent the queries, keys, and values, respectively, and d_k represents the dimension of the queries and keys.

Attention is then calculated at multiple instances to attend to different subspaces through multi-head attention: calculating attention with each instance's desired queries, keys and values and then concatenating the result in a single vector comprising the attention from all the heads.

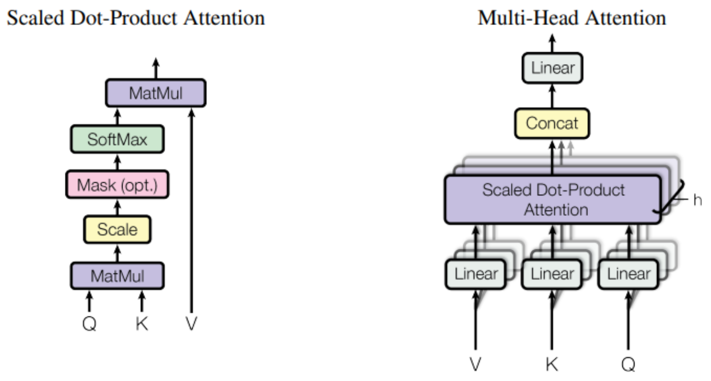


Figure 2.5: Visualization of scaled dot-product attention and multi-head attention. *Source:* [13]

In the image and vision processing field, attention has been included in CNN models in an attempt to have the model attend to the most relevant part of the input, and currently there are even models that are pure-transformer based, with the Vision Transformer[8] being the pioneer. It splits an image in fixed-size patches, linearly embeds them, adds position embeddings and feeds the result to a Transformer encoder.

Just like 3D-CNNs extend 2D-CNNs from image to video, Video Vision Transformer[14] extends Vision Transformer from image to video, by extracting tokens from the time, height and width dimensions.

2.7 RoIAlign

Many of the models in which our study focuses on use RoIAlign[15] - an operation that's very commonly used in computer vision related tasks, designed to accurately align features within the region of interest, hence the name (ROI). This tool allows to extract fixed-size feature maps for any given ROI size while accurately aligning the spatial features, by performing bilinear interpolation to compute the exact values of the input features at four sampled locations, and aggregate the result. For our problem in particular, there are always going to be ROIs at the players' locations, so RGB based solutions must input the features of those regions to the model, and RoIAlign is usually the chosen method.[16, 5].

2.8 Deep Learning Hyperparameters

DL hyperparameters are settings defined before training the model that may affect the success of the model. They do not change the architecture of the model itself, but will change how the training is done; therefore, they can be tuned to get the best generalized model possible for a given architecture.

2.8.1 Learning Rate

The learning rate (LR) dictates the size of the step taken towards the direction of the gradient descent, to minimize the loss function. The model's convergence speed and accuracy will depend on the LR, since if it's too high, the model can overshoot the optima, and if it's too low, it can get stuck in a local optimum.

Some optimizers, like the very popular Adam, already adjust their step size during the training process. Furthermore, other techniques can be used to change the learning rate and avoid the problems referred above, like using a learning rate scheduler that adjusts the LR as the training progresses, much like the Adam optimizer, or to split the training in two stages, where the first one has one initial LR and the second stage has a different starting LR.

2.8.2 Batch Size

The batch size (BS) defines the number of samples processed at each epoch during training. A large batch size will lead to a faster training, but comes with the downside of requiring more memory. The model can also converge better with a larger batch size as the gradient estimates can be more accurate this way, but this is generally not a concern.

2.8.3 Weight Decay

Weight decay is a regularization technique that aims to prevent overfitting and thus allow the model to generalize better. The concept is to add a penalty on the L_2 norm of the weights, with the intent of increasing the loss value for larger weights that often cause overfit, although if the weight decay is set too high, it can lead to underfit too. Mathematically, weight decay can be expressed as:

$$L_{new}(w) = L_{original}(w) + \lambda w^T w \tag{2.5}$$

2.9 Object Detection

Object detection (OD) is a computer vision task that involves detecting and classifying specific objects in images. The objective of OD is to output the bounding box of an object as well as the detected object's class.

In order to select the best detector, we must first understand how to measure their quality according to the expectations for the application in which GAR and HAR will be used.

For GAR, a missing actor in a group of 12, just like we often encounter in volleyball videos, may not have a large impact as the models can generalize in that sense for multiple reasons:

- Volleyball frames don't always contain the 12 players at once, so during the training the model will often have less than the total players in scene;
- Keypoint modality models may use actor dropout augmentation[2, 17], a data augmentation technique where a random actor is removed from a random frame. This will help the model generalize well in a situation where players are not detected, frequently due to player occlusions.
- RGB based models will take the missed player into account anyway, as they do not require player annotations or detections for GAR.

2.9.1 Mean Average Precision and Precision Metrics

Detection models are usually evaluated on their mean-average precision[18, 19, 20], which is expressed as in 2.6

$$\text{mAP} = \frac{1}{N_c}\sum_{i=1}^{N_c} \text{AP}_i \tag{2.6}$$

where N_c is the number of classes and AP_i the average precision for each class i. Hence, the commonly used metric mAP is correlated to the precision, a metric that evaluates only the correct predictions, meaning that if a bounding box is not predicted on an existing object, the precision will not account to that miss as it's a False Negative, as shown in equation 2.7.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{2.7}$$

In the context of object detection, True Positives (TP) represent the number of correctly detected objects and False Positives (FP) represent the number of incorrectly detected objects. Although not present in the precision metric, False Negatives (FN) would happen when the model fails to identify an object.

2.9.2 Recall Metric

For HAR, however, any application will be hindered by the missing player, as no model can classify an undetected player's action. Since accurate HAR is of interest for many of the possible applications for recognition in volleyball, we also need to evaluate the detection model's recall, a metric that measures how good the model is at not failing to detect an object. Unlike the precision metric, recall does take into account the false negatives, as shown in 2.8.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{2.8}$$

2.9.3 Area Under the Precision-Recall Curve and F1-Score Metrics

Now, if our application requires accurate GAR, we might want to use the detector with the highest precision, but if we also or exclusively care about action recognition we don't just want the model with the highest recall, as it could contain bounding boxes that do not contain players or any object of interest at all, so we must still take the precision value into account. For this we have two similar metrics: the precision-recall curve (figure 2.6), where the objective is to have the maximum area under the curve (AUC); and the F1-Score, a harmonic mean of precision and recall, expressed as 2.9 and seen in figure 2.7.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

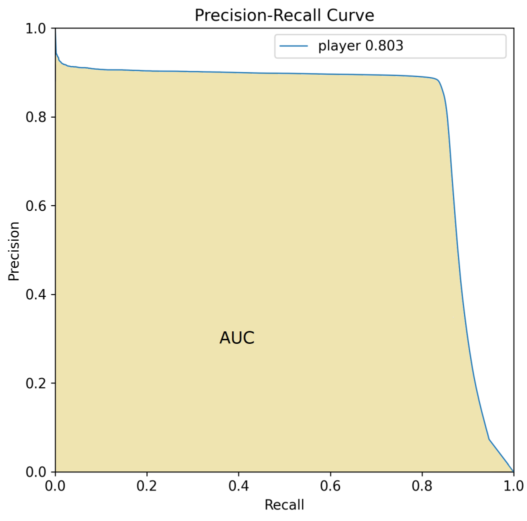


Figure 2.6: Precision-Recall curve with AUC.

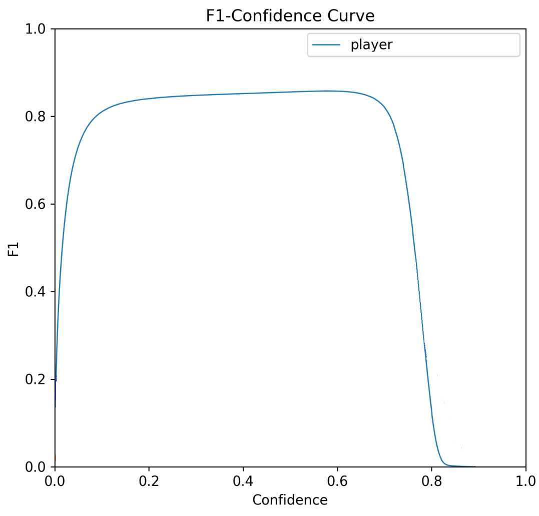


Figure 2.7: F1-Score across the conf. threshold.

More often than not, though, the best model, empirically observed as such, matches the one with the best F1-Score, as human observation will take into account both false positives, where a bounding box is drawn where no object of interest resides, and false negatives, where no box identifies an object.

In our detections we opt to evaluate based on the F1-Score of the model, because it's a better match with empirically observed overall quality of the detector and it's easier to see what confidence threshold we should choose to have the best possible precision and accuracy.

2.10 Multiple Object Tracking (MOT)

For video classification where multiple objects of the same class can be seen simultaneously, object detection and multiple object tracking go hand-in-hand. Having done a previous detection of the actors, it becomes necessary to associate each actor's bounding box to the corresponding identifier. Therefore, object tracking is critical in video group activity classification, as it's important to know each player's spatio-temporal variations, and for that, each player must be identified at all times.

Object trackers attempt to predict the trajectory of an object[21, 22], and thus finding the object in the following frames. Then, this prediction is compared to the bounding boxes detected beforehand and to each box is attributed its ID.

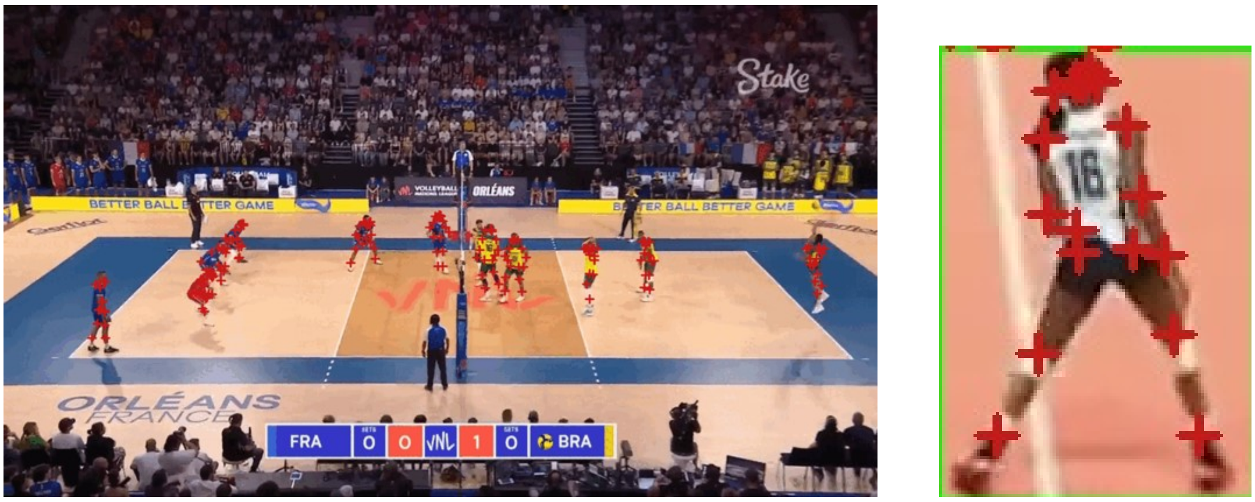
2.10.1 Multiple Object Tracking Accuracy (MOTA)

Usually, tracking models are evaluated on their MOTA, a metric that weighs in false positives, mismatch errors and misses. In the context of MOT, False positives occur where a prediction where no object is present is made; mismatch errors are incorrectly matched boxes, that can happen when objects are occluded for some frames; and misses, or false negatives, where an object is not hypothesised by the tracker.

2.11 Keypoint Detection

Keypoints are interest points in the image. For action recognition, the keypoints are represented by the individual's key body joints, such as shoulders, elbows, wrists, hips, knees, and ankles, as shown in figure 2.8. Keypoints may promote a model that generalizes better to any scenario as they eliminate any background noise.[2]

There are very efficient pose estimators available to detect keypoints from images or from previously detected or annotated regions of interest. These pose estimators usually follow the COCO (Common Objects in Context) dataset baseline keypoints, that can be seen in figure 2.8b.[23, 3, 2]



(a) (b)

Figure 2.8: (a) Example of keypoints extracted with the HRNet model from a volleyball video frame. (b) COCO keypoints in a volleyball player. The 17 keypoints include 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.

Since keypoints are not annotated on the VD, we must use some kind of metric as input to the model along with the keypoint's default features, that already include their coordinates and the type (e.g. left shoulder). This metric is introduced with the intuition of having a weight embedded in the keypoints that allow the model to generalize better despite inaccurately detected keypoints.

2.11.1 Object Keypoint Similarity Metric

The OKS metric is commonly used in the COCO dataset for evaluating the accuracy of keypoint predictions. It measures how far off the predictions are from the ground truth with the Euclidean distance.

$$\text{OKS} = \frac{1}{N}\sum_{i=1}^{N}\exp\left(-\frac{d_i^2}{2s^2k_i^2}\right) \tag{2.10}$$

where, N represents the total number of keypoints. For each keypoint, d_i is the Euclidean distance between the ground truth keypoint location and the corresponding predicted keypoint location. The value of s is a scaling factor, and k_i is a per-keypoint constant that depends on the size of the object instance, for example, according to the table 2.4.

Table 2.4: k_i for each type of COCO keypoint.

Kpt	Hips	Ankles	Knees	Shoulders	Elbows	Wrists	Ears	Nose	Eyes
k_i	0.107	0.089	0.087	0.079	0.072	0.062	0.035	0.026	0.025

When the predicted keypoints are very close to the ground truth keypoints, the similarity score given by the OKS is close to 1 and on the other hand, when the detected keypoint is far from the ground truth, the OKS is close to 0. Since the equation involves an exponential function, large errors are penalized heavily, while small errors are taken more lightly.

3 Overview of Action and Group Activity Recognition

3.1 Models Benchmarked on the Volleyball Dataset

As mentioned in the previous chapter, the Volleyball Dataset is widely used as a benchmark dataset for group activity recognition, so there are many models designed for this task that have been trained and tested on the dataset.

3.1.1 Hunting Group Clues with Transformers for Social Group Activity with Transformers[24]

The current state of the art in group activity recognition for the Volleyball Dataset GAR task achieves 96.0% for GAR and 65.0% for HAR in Tamura et al.'s Hunting Group Clues with Transformers for Social Group Activity Recognition where transformers are used to generate social group features in an attempt to use attention modules to better understand the context of the social scene. It's important to note that in order to achieve state of the art results in group activity recognition in the Volleyball dataset, Tamura et al. also used annotations on group member information provided by Sendo and Ukita[7]. Furthermore, the data is augmentated randomly using multiple data augmentation techniques.

The work developed by Tamura et al. uses the RGB stream of I3D as the backbone feature extractor and feeds input features from 3 layers to the deformable transformers, that are trained with parameters trained on the COCO dataset, and I3D is initialized with the parameters trained on the Kinetics dataset. I3D (Inflated 3D ConvNet) is a convolutional neural network that expands ConvNets to 3D, incorporating spatio-temporal features extractors in videos[25].

Although this transformer based architecture achieves the state of the art for the Volleyball Dataset GAR task, its accuracy of 65.0% in the HAR task puts it far behind other models that surpass this value by over $\sim 10\%$ while closing the gap in GAR accuracy, being only worse by less than $\sim 3\%$.

3.1.2 COMPOSER[2]

The COMPOSER model, introduced by Zhou et al. also achieves great accuracy for the group activity recognition with 95.00% and 71.4% for HAR utilizing multi-scale transformers. In the COMPOSER's paper, a keypoint-only modality is proposed, in order to avoid biases from different scenes. The paper also utilizes another data augmentation technique besides the typical random horizontal and vertical flips and rotations: the actor dropout; first proposed by Ngiam et al.[17] where a random actor in a random frame is removed.

The COMPOSER's architecture takes as input keypoints. There are 4 scales, of which the first consists in encodings of the keypoints, covering coordinate, time, type and OKS-based feature embeddings; the second is derived from the first one by aggregating each person's keypoint through a multi layer perceptron, creating a representation of each person; following the same logic comes the third and fourth scales, where the third one aggregates persons in person-to-person interactions and the fourth aggregates persons in groups. These 4 scales and their classifier token (CLS) are input to 4 corresponding transformer encoders: at each scale, the transformer takes as input the scale's tokens and the previous transformer's encodings. The architecture can be best understood in figure 3.1

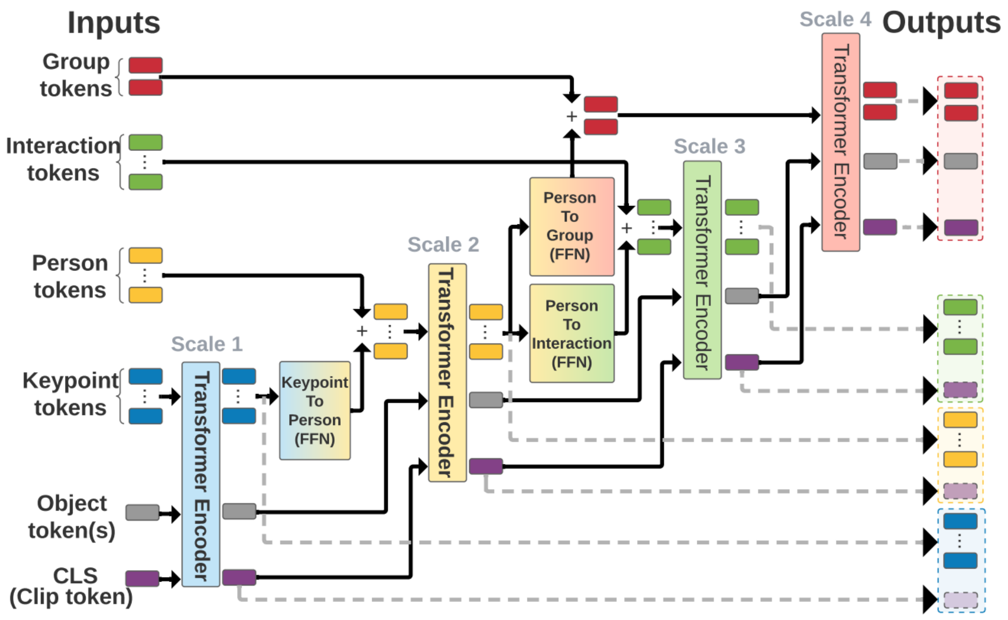


Figure 3.1: COMPOSER's multiscale transfomer architecture. *Source:* [2]

Agreement between all the scales is necessary, therefore, the clip representations learned in each scale are clustered and a swapped prediction mechanism is used to predict one scale's cluster assignment from another scale's representation. Meanwhile, each scale's output is fed into a group activity classifier while the person scale output is also fed into a person activity classifier. Then, when calculating the loss, the swapped prediction problem's loss is added to the sum of the cross-entropy loss of each scale's prediction of the GAR as well as the person scale prediction of each HAR.

3.1.3 Spatio-Temporal Dynamic Inference Network[16]

Just as the other RGB based models, the Dynamic Inference Network (DIN) starts by extracting visual features from the input video clip. RoIAlign[15] is then used to extract each individual's features aligned with the annotated or detected bounding boxes. These individual features are ordered temporally and spatially to be input to a Spatio-Temporal graph that models the interactions between players. This process shall result in a total of $T \times N$ graphs, where T is the temporal dimension (effectively the number of frames used) and N the number of actors annotated. The architecure can be seen in figure 3.2 and the graph construction is detailed below.

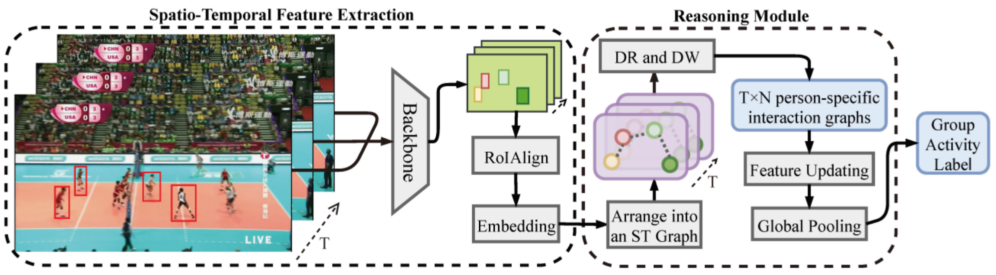


Figure 3.2: DIN architecture. *Source:* [16]

The graphs themselves are dynamically predicted, hence the name of the model, by Dynamic Relation (DR) and Dynamic Walk (DW): two modules used to predict one person's interactions and arrange them to a representative graph. DR predicts the relation matrix of each person, a representation of the interactions between actors, by being applied to an interaction field: a spatio-temporal field around the person that should contain the persons that can be interacted with and will therefore appear in the interaction graphs. DW is then applied to predict offsets within the field, modeling the spatio-temporal dependencies; since DR predicts the representation of the interactions, they are not predefined anymore, thus creating the need to endow the interaction graph with a global interaction field, solved by the DW module.

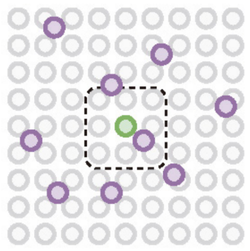


Figure 3.3: Dynamic graph inference. The green node represents the central person, the dashed box the initialized interaction field and the purple nodes the features related to the green node's changes over space and time.*Source:* [16]

DIN reports an accuracy of 93.6% on the VD GAR task, with a VGG-16 backbone. However, there are no reported HAR metrics for this model, even though the model also predicts person actions in parallel to the activity classification. The authors claim that action labels are incorrectly annotated which justified not using a cross-entropy loss for individual actions alongside the cross-entropy loss for GAR.

3.1.4 DECOMPL

DECOMPL[5] is an RGB only model, as its input consists in just the frames. Like DIN, DECOMPL uses a backbone CNN to extract RGB features from the frames and RoI align to have bounding boxes coordinates. The classification task is separated in two branches: the visual branch, that processes the visual features of the image and the individual bounding boxes; and the coordinate brach, that processes the box coordinates, modeling the interactions between them. Attention pooling is then used in both branches separately.

The visual branch uses a VGG backbone, just like the previously described DIN model, to extract visual features. This is applied to the image itself and to the alignments of the bounding boxes. Based on their bounding boxes' coordinates, the players are split in left and right team: two sublists of features that are fed into the multi-head attention pooling (MHAP) network.

At the same time, the coordinate branch processes the layout of the players in the scene and aims to represent the relative positions of the players in the frame. For this reason, the features are not split in left and right before the attention pooling this time.

The attention pooling mechanism is chosen due to being trainable and due to its robustness to permutations, that happen very frequently in the context of players not taking the same positions in every instance of the same activity. Differently from average or max pooling, attention pooling aims to pick the most relevant set of features through the calculation of self-attention, instead of a fixed magnitude (average or greatest values).[26]

Afterwards, the classification task is split in three different labels: side, team activity and group activity (the final prediction). This evaluates and calculates loss with respect to the side label (right or left), the team activity label (pass, set, spike, winpoint) and the group activity label (r-pass, l-set, etc). Both the predictions of the visual branch and the coordinate branch weigh in on the final predictions for each classification subtask.

The described architecture of the described DECOMPL model can be seen in figure 3.4.

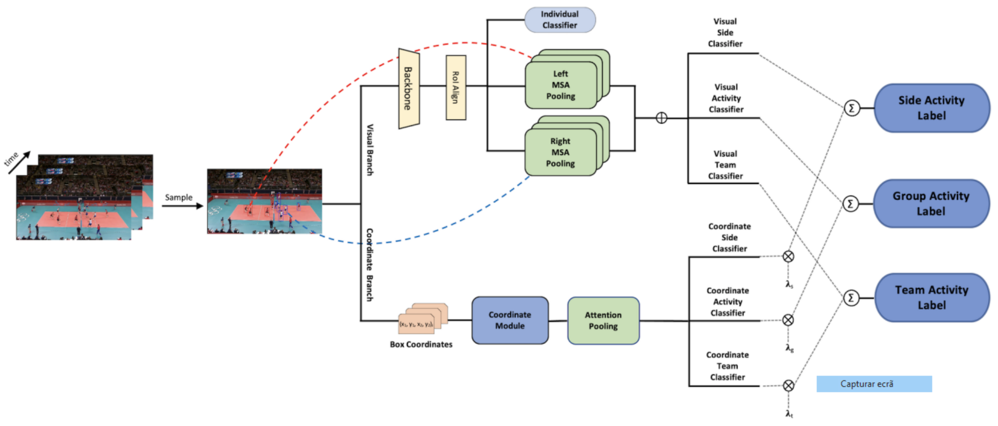


Figure 3.4: DECOMPL architecture. *Source:* [5]

3.1.5 GroupFormer

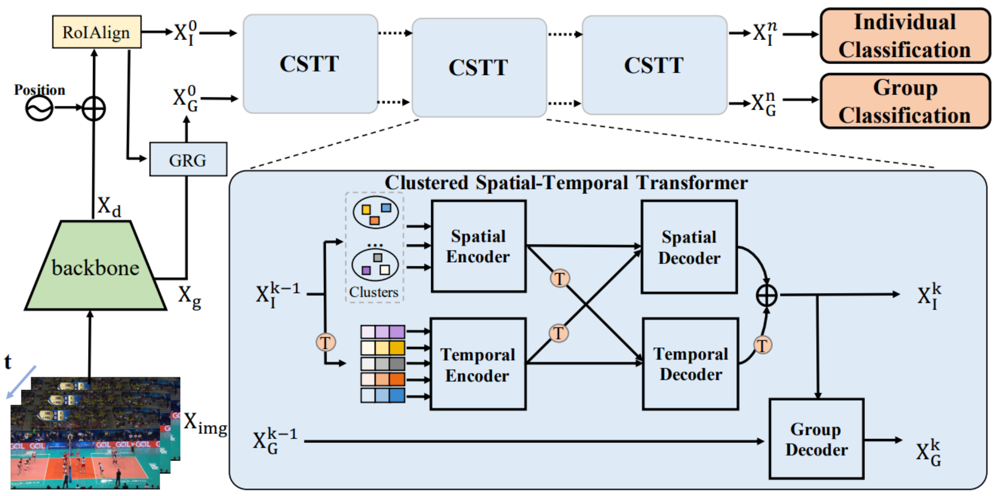


Figure 3.5: Groupformer's architecture. *Source:* [27]

Groupformer achieves a GAR accuracy of 95.7% and a HAR accuracy of 85.6% with clustered spatial-temporal transformers (CSST). As shown in the architecture in figure 3.5, the frames are first processed by a CNN backbone, I3D, and a Group Representation Generator, that creates an aggregation of scene and actor features extracted with RoIAlign[15]. Individual features and group representation are then fed to the clustered spatial-temporal transformer for predicting individual action and group activity.

The CSTT first has a spatial and a temporal encoders that generate spatial and temporal features in parallel. Then, the outputs of both encoders are fed to two decoders: a spatial decoder, where the spatial transformer output is seen as the query and the temporal transformer's output is the key and value, and a temporal decoder where the spatial embedding is regarded as the key and value while the temporal embedding is the query. At last, a Group Decoder takes the individual representation and the group representation, output from the individual decoders, to aggregate actors' interactions and the whole multi-person scenario.[27]

3.2 Action Classification in Video

Most datasets do not contain multiple people performing actions simultaneously in a cluttered environment like a volleyball game. Action recognition datasets like the Kinetics 400[28] usually contain very distinct, sparse actions like braiding hair or stretching, that can take several seconds, in some cases even a few minutes.

Still, because they are the SOTA for action video classification it's worth to mention TubeViT[29] as a potential model to use in our GAR and HAR task. TubeViT applies the concept of sparse video tubes: the video is tokenized sparsely, by having each kernel with a large temporal stride and varied spatial strides. A spatial encoder, based on a vision transformer architecture where the frames are divided into non-overlapping patches and each patch is embedded into a token using self-attention mechanisms, is used to process each individual frame in the video. Then, the spatial embeddings of consecutive frames are used to model temporal relationships between frames by being input to a temporal encoder that applies a layer of self-attention across the time dimension.

In the volleyball HAR context the actions at hand do not meet this criteria; they are rather dense actions, e.g. actions are similar to each other and occur in very short time periods (40 frames for the VD specifically). Actions with these same features can be found in other competitive sports, for instance.

3.3 Action Recognition in Sports

Wu et al.[1] have done a survey on video action recognition in sports, which is of great relevance for this work's research, as it includes not only results, methods and datasets for volleyball, but also for a plethora of other sports. Their paper approaches all kinds of methods using 2D features, 3D features, two-stream features and skeleton features. They report that, lately, convolutional kernels have started to be replaced by multi-head self-attention, in a combination of both spatial and temporal self-attention, which leads to outperforming previous architectures on some action recognition datasets. Due to spatial and temporal modules, 2D deep modules are increasingly adopting all kinds of transformers along with pre-training with larger datasets in order to train spatial networks.

4 End-To-End Volleyball video GAR and HAR

4.1 Methods

For actual video processing where actions are detected across the volleyball clip, it's necessary to pre-process the video frames to have all the data that's needed as inputs to the chosen model. Most models will require at least player bounding boxes information as input, so if we want to input only the frames to a model, a detection phase must be implemented in the pipeline before applying the GAR and HAR model. Since each player needs an ID to keep continuity of individuals in the temporal context, a tracking solution has to be implemented alongside the detection.

The developed pipeline architecture receives as input a video and after turning it into frames and organizing them in folders for sampling, starts by detecting all the players with an object detector, moving on to tracking the detected players and to detect the keypoints if necessary; finally, the appropriate modality (bounding box coordinates or keypoint) is fed to the chosen GAR classification model, as can be seen in figure 4.1.

Due to the paradigm created by the VD as a benchmark baseline for GAR, every tested model expects a fixed number of frames as input, where the central frame is the one where the actions and activity occur. By uniformly sampling frames from the full video we will end up with frames that don't necessarily meet these conditions. Even if some of the frames are timed correctly within a reasonable margin, the activities don't always take the same time to complete which will lead to further errors, so we also study how robust the models are for these kinds of real scenario conditions, where the time sequence is not centered on the central frame, because the frames are being uniformly sampled without supervision.

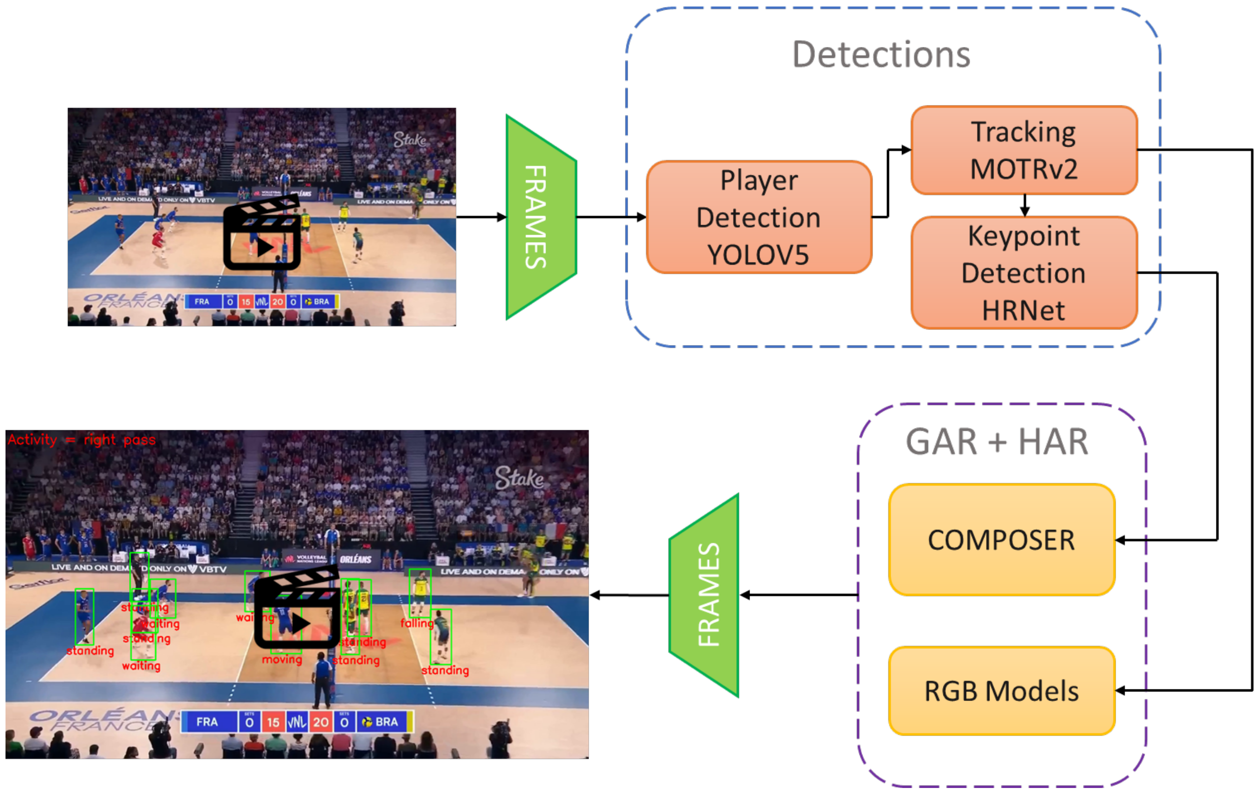


Figure 4.1: Architecture of the full continuous action and activity classification for volleyball videos.

Further in this section we go deeper into each studied model for all 3 concerned tasks: OD, MOT and GAR and HAR. We study their methods, results and the ease of use and deployment for larger projects.

All the tasks and models described below are validated and tested in the VD and further tested on a very small (632 frames - 30 video segments) unannotated test dataset, created from the Brazil vs. France game of the Volleyball Nations League 2023, made available by Volleyball World at Youtube, which should be enough to test the models in an environment that is different from the VD. Because this video was not annotated, all we can infer from the tests conducted on it comes from empirical observation and therefore can't be unaccompanied by the metrics measured on the VD test split.

4.1.1 Player and Ball Detection

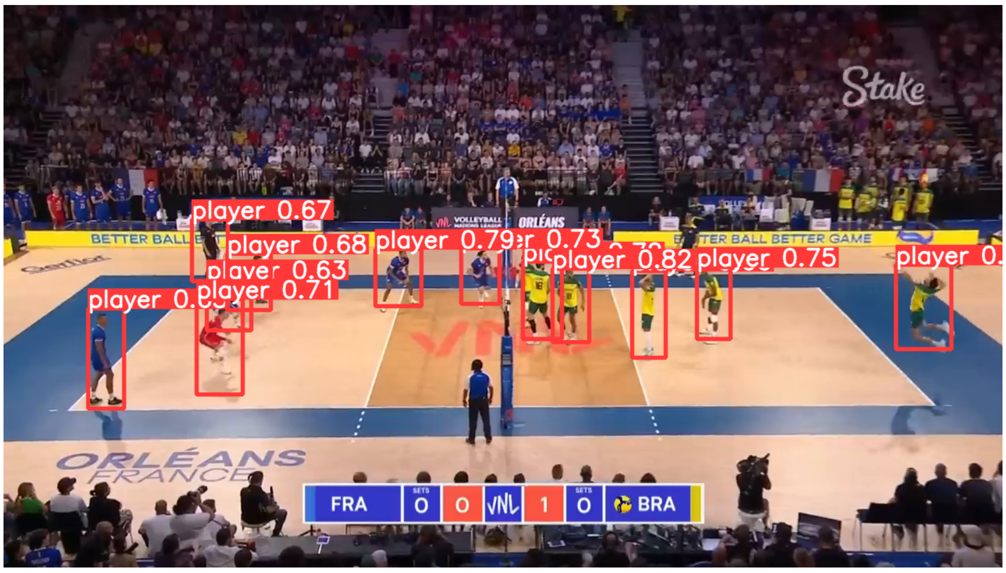
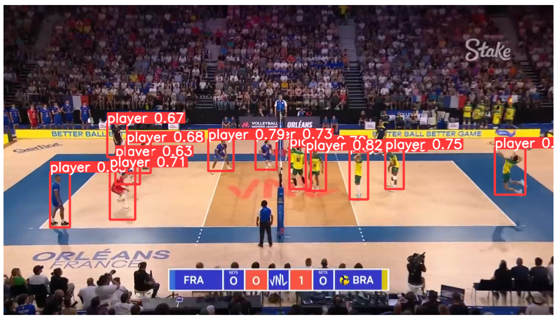
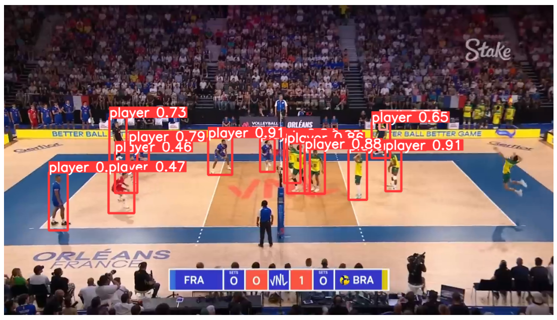


Figure 4.2: Yolov5 player and ball detection results on test data. The results are good enough for GAR as shown in further chapters, even though there is a false positive as the referee is detected and a false negative where an occluded player near the net is not detected

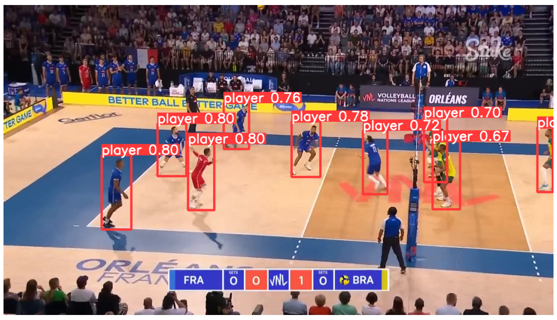
Player bounding boxes can be quickly and effectively obtained with YOLO Networks[19]: single-stage object detectors whose recent versions (YOLOv5, YOLOv6, YOLOv7) keep achieving SOTA precisions in object detection tasks[30]. For all the tested models we initialize the weights with the YOLOv5-x model, the most complex of the available weights, since this is what leads to the best results within the YOLO models.



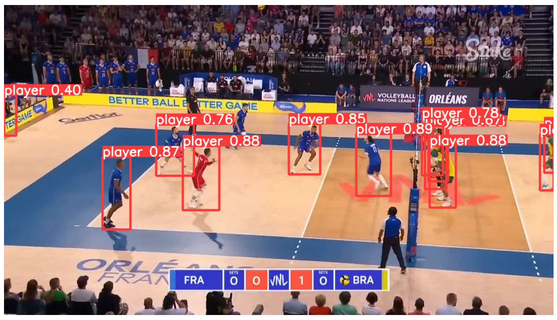
(a) YOLOv5



(b) YOLOv8



(c) YOLOv5



(d) YOLOv8

Figure 4.3: (a) and (c) YOLOv5 detections in the first and last frames of an activity. (b) and (d) YOLOv8 detections in the same frames. None of these frames are perfectly detected, as the referee is detected in (a) and (b); (b) is missing the rightmost player; (c) is missing an occluded player near the net; (d) misses one player, has a duplicate detection on the rightmost player and the leftmost box is incorrectly detected.

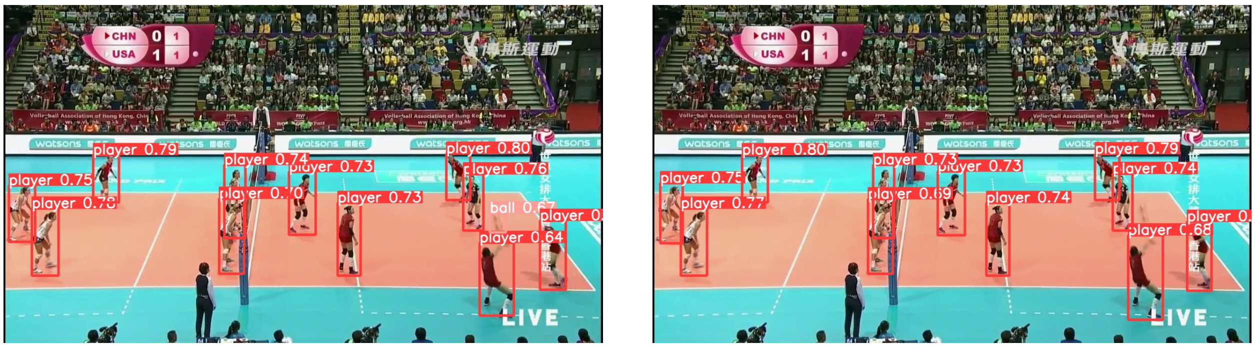
Table 4.1: Comparison between YOLOv5 and YOLOv8 metrics.

YOLO Network	**Precision**	**Recall**	**F1-Score**
YOLOv5	0.81	0.71	0.75
YOLOv8	0.81	0.64	0.71

Furthermore, when empirically observing the detection results in unforeseen data (figure 4.3), YOLOv5 seems to detect players more accurately, even if YOLOv8 detections are also very much acceptable. For these reasons we choose to implement YOLOv5 as the detector to use. Tests and results with both YOLOv5 and YOLOv8 networks are presented in appendix A.

Although YOLO detections are not perfect, we will later see that tracking can help mitigate this issue, as long as the players are only incorrectly detected in a few frames of the total sequence.

Ball detections, on the other hand, are not reliable to input to a HAR/GAR model. This happens because the ball is often blurred due to the low frame rate at which the video was recorded combined with the great velocities that the ball moves in as shown in figure 4.4. This is not an issue that compromises the functioning of the pipeline, as no model available depends heavily on ball position, with most tested models not using ball annotations at all.



(a) (b)

Figure 4.4: (a) YOLO accurately detecting a blurred ball (near the group of players on the right). (b) YOLO missing the ball on the immediate next frame. Although not showed here, the next frames also don't have the ball detected by YOLO.

YOLOv5 Implementation Details

We train the available YOLOv5 network on the VD following the same split that is suggested by the authors of the dataset. With the weights initialized by the available checkpoint from the training on the large COCO dataset, we train over 5 epochs with a learning rate of 0.01 and a weight decay of 0.0005, with images of size 1280×720. For inference, we set the confidence threshold to 0.5.

After detecting the bounding boxes for inference, we store the results in a pickle file, with the bounding boxes in the YOLO format: $[x_{center}, y_{center}, width, height]$, which differs from the format of the input data of the models trained on the VD, but we solve this issue after completing the tracking association on the detected bounding boxes, which also requires annotations in the YOLO format. By having this file, we can easily run tracking and GAR and HAR inference with different models without having run YOLO detection once more.

This format mismatch has also created the need to transcribe the VD tracking annotations to the YOLO standard to make it possible to train the network in the dataset.

4.1.2 Player Tracking

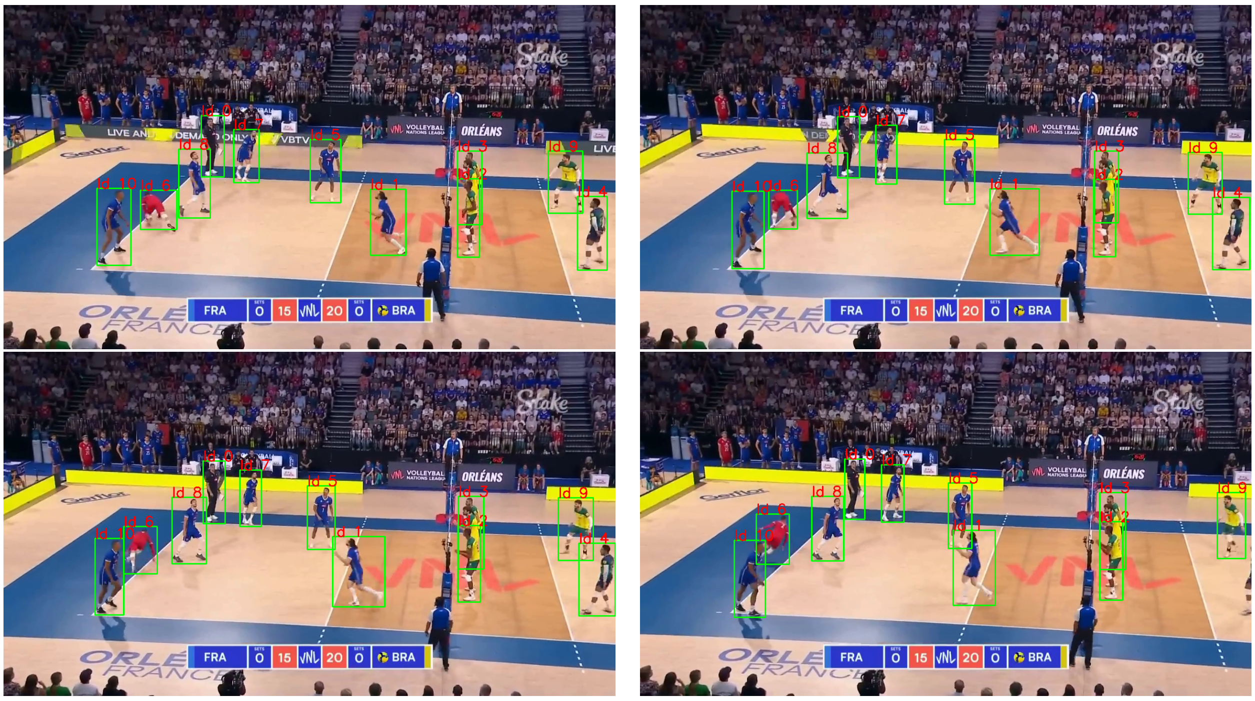


Figure 4.5: Tracking performed by MOTRv2. For better visualization, the frames were sampled with a sampling frequency of 4 and are temporally ordered from left to right. We can see that the associations are very successful, even though the referee is detected in all frames because he stands very close to the players, which does not happen in the training conditions of the VD. (These frames do not belong to the VD)

For all classification models it's key to have not only players bounding boxes, but also IDs associated to their bounding boxes that identify each unique player in all frames, as seen in figure 4.5. Hence object tracking also belonging in the end-to-end frame to HAR/GAR pipeline, jointly with player detections. Two tools were tested for this task:

- ByteTrack - Tracking method integrated in most recent YOLO networks, so it's very practical to use right after applying YOLO inference for player detection. ByteTrack uses BYTE, a data association method to first associate the high confidence detection boxes to the tracklets, and then also associate the lower confidence detection boxes to the previously unmatched tracklets.

  ByteTrack is originally created by combining the detector YOLOX[31] and the association method BYTE[21].

- Motrv2 - Achieving SOTA MOT results in the DanceTrack dataset[32], Motrv2 is based on MOTR[22], a tracker built on the Deformable Transformer architecture, introducing the track and object queries. The track query is initialized with the output of the object query of the detected objects and updated over time to predict tracks. MOTR had the inherent problem of end-to-end MOT frameworks, which is poor detection performance, unlike their counterpart, tracking-by-detection. Thus, in the official paper, MOTRv2 uses YOLOX to perform detections on all frames.

In Motrv2, track queries are generated based on the prediction of the previous frame and may be assisted in detection by the YOLOX proposals, while the detect queries are actually replaced by proposal queries, that aggregate information from the track queries. As seen in figure 4.6, proposal and track queries must both be aggregated to accurately detect and identify the actors while avoiding duplicate detection.[33]

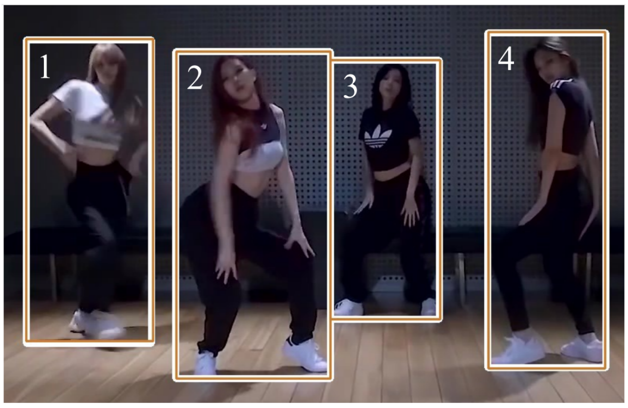
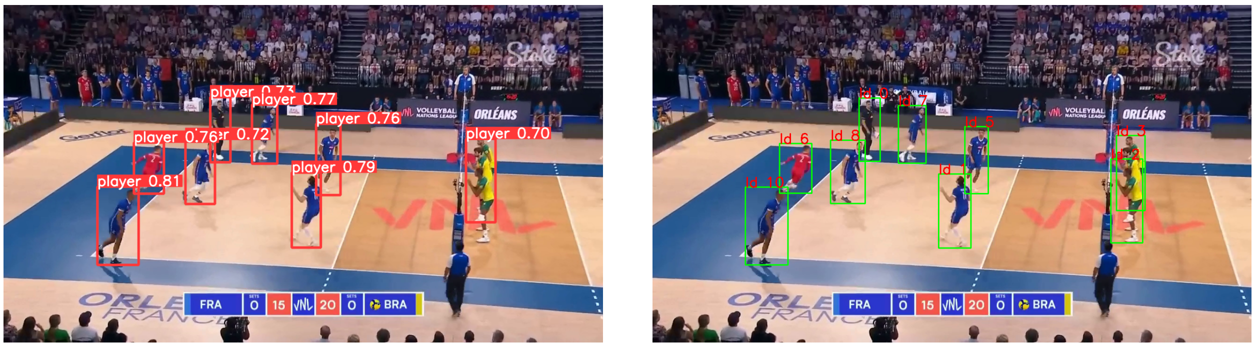


Figure 4.6: Visualization of track query prediction (brown box) and proposal query given by YOLOX detection (white bold box) in the MOTRv2 model. *Source:* [33]

Because of the track query prediction, it's possible to improve the prior detections, given by the detector in the case of a false negative, where a player is not detected in one of the frames, but it's possible to predict its position through the previous frames' detections, practically increasing the recall of the model. However, this comes with the cost of reducing the precision of the model, as in some cases, mostly when the referee is detected in some frames by the detector, it is also then detected through the whole sequence by the tracker. By our empirical analysis, since the referee should never have a relevant action and is not too far from the group when detected, the GAR task does not suffer from their detection and the HAR will usually be classified as standing, which is also not critical to the understanding of the clip's context. Thus, it should be preferred to detect all the players and the referee than to not detect the referee while not detecting all the players, potentially missing a player performing a relevant action, like passing or digging.

Overall, ByteTrack achieves great results, but because it's not implemented to detect predicted player's bounding boxes like MOTRv2 by default, we opt to use MOTRv2 in our implementations. By replacing the YOLOX detector by our YOLOv5 weights trained on the VD we can make use of MOTRv2 not only for tracking but also for assisting the player detection, as seen in figure 4.7



(a) (b)

Figure 4.7: (a) YOLO detection misses the player with tracking ID 2 (on the right), that is actually detected in (b) MOTRv2 tracking results.

MOTRv2 Implementation Details

The MOTRv2 available weights were obtained from training on the DanceTrack dataset[32]. This dataset contains video of groups of people dancing recorded from a static camera, which resembles the tracking task on volleyball video, where players are frequently scrambled but the camera angle remains similar through the whole frame sequence. Therefore, specific training was not necessary, as those weights are enough to achieve a MOTA of 92.1% on the DanceTrack dataset and the results carried over to the VD.

We input the detections from the YOLOv5 network trained on the VD for the detection task to the MOTRv2 inference module to perform association between each detected box in each frame.

We make changes to the available inference code in order to run in our video whose frames should be split in folders in order to be input to the models trained on the VD without changing the models' dataloaders. To do this, we simply call the implemented *detect* module and reset the IDs at each call, so that every player ID is always between 0 and 11. Sometimes it may happen that the player's ID is different between two sequences of frames (for example, folder with frames 1-40 and folder with frames 41-80 may have player's IDs switched), but this is not an issue as the GAR models are ran independently from each sequence and the player's position is taken into account, its ID isn't but a measure of continuity.

After running the tracking inference, the results are saved in a pickle file as a dictionary that can be represented as $tracks[(video, instance)][frame] = np.array(12, 4)$ where each array represents the corresponding player ID (the array in the index 0 is the player identified as $ID = 0$ and so on until the 12th player ($ID = 11$)) and contains normalized bounding box coordinates as $[y_{min}, x_{min}, y_{max}, x_{max}]$, that can be understood in figure 4.8. This file is thus created in accordance to the format of the tracking annotations of the VD and can be used to input the bounding box coordinates to the models that were trained on the VD correctly.

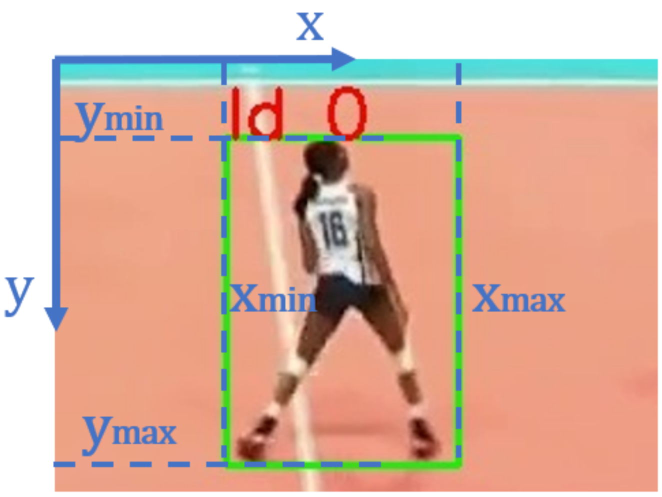


Figure 4.8: Visualization of resulting bounding box coordinates from tracking.

4.1.3 Keypoint Detection

As previously mention in chapter 2 we use HRNet to extract the keypoints from the video frames. As mentioned in the same previous chapter, we also want to add an OKS metric value to the keypoint data. In the COMPOSER architecture, the OKS metric is calculated between frames and input to the model alongside the keypoints' coordinates. Because we don't have ground truth annotations for keypoints, this is the solution the authors came up with to deal with noisy keypoints, as between two successive frames the distance between the same keypoint is expected to be quite small: and sometimes keypoint detections can indeed be very inaccurate, as seen in figure 4.9.

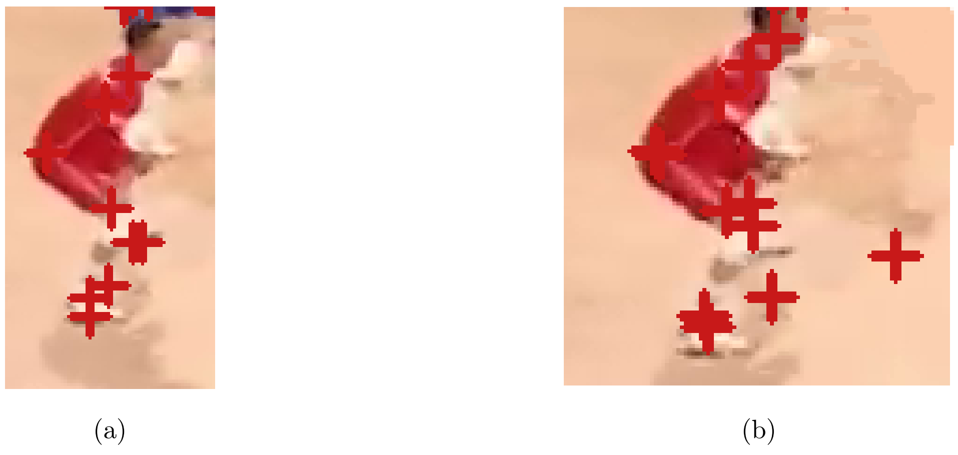


Figure 4.9: (a) An accurately detected keypoint (b) The same keypoint inaccurately detected in the next frame.

To detect keypoints, we must input the bounding boxes coordinates for the HRNet to apply RoIAlign and detect the keypoints in the players' bounding boxes. For faster inference, we could not input the bounding boxes and let the network detect keypoints in all detected persons, but some of them would be referees and spectators, which we do not want to be included in the actor groups for GAR and action classification.

As seen in the previous section, object tracking can improve OD, therefore the boxes we input to the HRNet for keypoint detection are the ones resulting from the MOTRv2 tracking, and not the YOLOv5 detection.

4.2 Tested Models Results

In this section we dive into the implementation details and results achieved with the available and previously mentioned in 3 methods on the VD itself. For all of them, we use the publicly available code and weights when applicable, and for the ones that achieve the best accuracy, namely DIN, DECOMPL and COMPOSER, we develop the necessary code to run the inference on new data.

4.2.1 Groupformer

Although Groupformer reports great results in both GAR and HAR tasks, the available code at `https://github.com/xueyee/GroupFormer` seems to not be up to date and missing critical architecure related files so it's not possible to test the model or to use it for inference. We tried to contact the authors about these issues, but to no avail.

Furthermore, even if they do not detail the reason why, Demirel and Ozkan (DECOMPL), also exclude GroupFormer from their comparative analysis as they report that they couldn't reproduce the results.

4.2.2 TubeViT

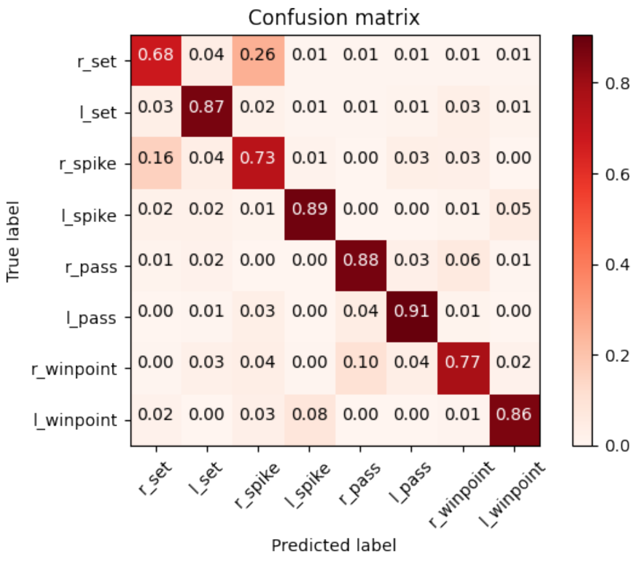
As mentioned in 3, TubeViT is widely used in video classification tasks, and even achieving SOTA accuracies. However, the benchmark datasets in which TubeViT is tested, like the Kinetics dataset, aren't too similar to the context of a volleyball game. The videos in the dataset are usually longer (with a duration of at least 10 seconds) than the duration of an activity in a volleyball game (around 2-3 seconds) and the videos that contain timesiple actors have them all performing the same action as a crowd. The fact that there are crowds performing the same action, meaning a crowd has one label only, could lead to the idea that the model would also perform well on group activity, since there is only one label too, even though the players act in a group instead of independently performing the same action.

In reality, the TubeViT model did not achieve good results at all in the VD GAR task, in which there is a great overfit: the loss function ends up converging during training, but in validation the GAR accuracy does not go beyond 25%.

The model was implemented with the Adam optimizer, a BS of 16 and a LR of 0.00005. We used kernels of size $8 \times 32 \times 32$, $4 \times 128 \times 128$ and $12 \times 256 \times 256$, with strides $8 \times 32 \times 32$, $4 \times 128 \times 128$ and $1 \times 256 \times 256$, respectively.

4.2.3 Dynamic Inference Network

We train the DIN model on 720×480 images, with the Adam optimizer, a LR of 0.0001 dropping by a factor of 2 at every 30 epochs. We train for 120 epochs and obtain the results shown in figure 4.10. Since the model does not use action label supervision, we do not evaluate on the individual action, just like the original paper.



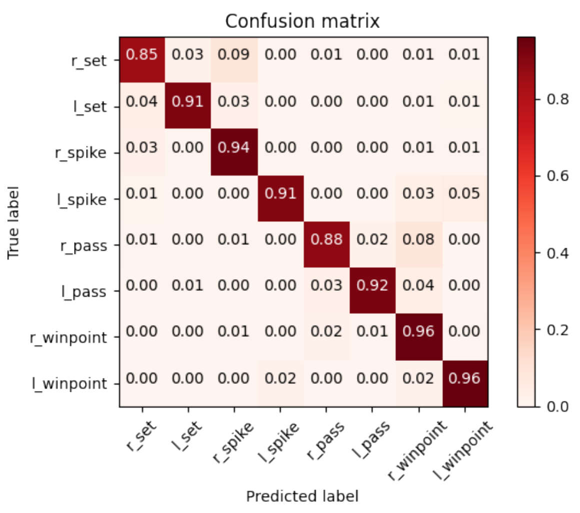
	r_set	l_set	r_spike	l_spike	r_pass	l_pass	r_winpoint	l_winpoint
r_set	0.68	0.04	0.26	0.01	0.01	0.01	0.01	0.01
l_set	0.03	0.87	0.02	0.01	0.01	0.01	0.03	0.01
r_spike	0.16	0.04	0.73	0.01	0.00	0.03	0.03	0.00
l_spike	0.02	0.02	0.01	0.89	0.00	0.00	0.01	0.05
r_pass	0.01	0.02	0.00	0.00	0.88	0.03	0.06	0.01
l_pass	0.00	0.01	0.03	0.00	0.04	0.91	0.01	0.00
r_winpoint	0.00	0.03	0.04	0.00	0.10	0.04	0.77	0.02
l_winpoint	0.02	0.00	0.03	0.08	0.00	0.00	0.01	0.86

Figure 4.10: Confusion matrix for GAR with the DIN ($accuracy = 88.37\%$).

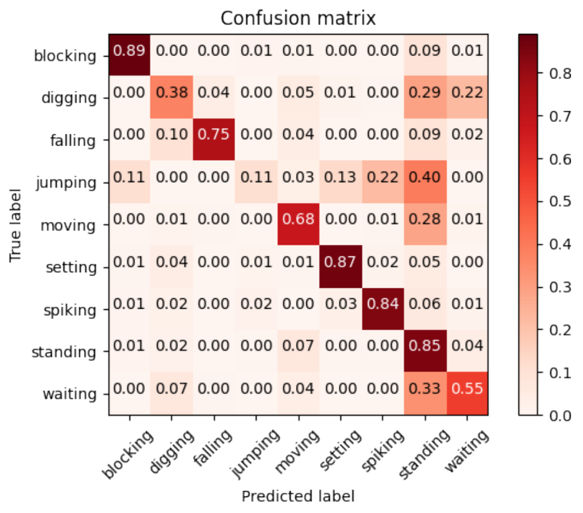
We see a decline in performance comparing to the accuracy of 93.6 that is reported on the paper. This can be caused by the lower resolution that we train on due to size constraints, as we could only train on 720×480 images, which would lead to worse results than training on the 1280×720 images like on the paper.

4.2.4 DECOMPL

We train the model as it's available, on 640×360 images, with the Adam optimizer, a learning rate of 0.0001 dropping by a factor of 2 at epochs 30, 60 and 90 for a total of 120 epochs: all the hyperparameters as described in the original paper. We also use the same backbone (VGG16) and the same batch size of 8. The results on the VD split show 91.62% GAR accuracy and 65.7% HAR accuracy, as seen in 4.11.



(a)



(b)

Figure 4.11: (a) confusion matrix for GAR with the DECOMPL ($accuracy = 91.62\%$) and (b) confusion matrix for HAR with the DECOMPL ($accuracy = 65.77\%$)

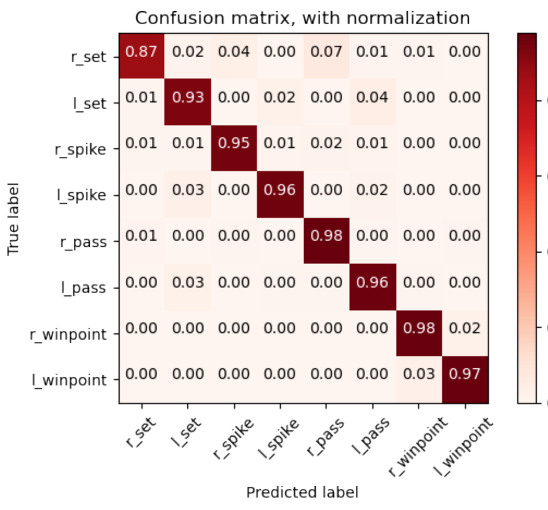
We also implement actor dropout augmentation in DECOMPL training, but this led to an accuracy of 75.77% for GAR on the VD train-test split.

From the confusion matrices in 4.11, we can see that every activity is well predicted, given the 91.62% accuracy achieved. Every class except right set and right pass are predicted with an accuracy over 90%, which we later see that actually occurs because of the incorrect annotations of the VD. On the other hand, the action recognition accuracy is not as good, which happens for timesiple reasons: the standing action is a lot more frequent than the others which can cause the model to overfit to predict that class, even though the actions weights are initialized according to the imbalance. Also, since the model is designed to classify group activity and not specifically short duration human actions, not only is its architecture optimized in that direction, the loss function is also designed accordingly, as it weighs in the side of the activity (*e.g.* left), the activity without the side (*e.g.* set) and the activity and side as annotated in the VD (*e.g.* left-set), while only weighing the action classification. Furthermore, the action labels are inaccurately annotated in the first place, as mentioned previously in chapter 2.

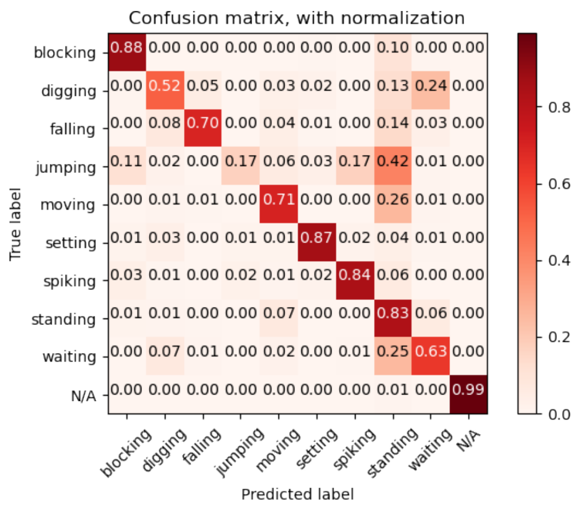
4.2.5 COMPOSER

The COMPOSER's trained weights are made available by the authors, along with the official implementation of the code. The resulting confusion matrix of the available weights can be seen in 4.12. These results are also reproducible by training the model with the configuration files annexed with the available code, with 10 frames uniformly sampled over two training stages as descrbied in the paper: a first one with the Adam optimizer, a LR of $5e-4$ and a weight decay of $1e-3$ over 40 epochs; and a second one with the same frames as the first one, the Adam optimizer, a LR of

$1e-4$ and a weight decay of $1e-3$ over 5 epochs. The second stage must start with the weights produced by the first stage. Some more tests were conducted with different hyperparameters on the COMPOSER model and their results and implementation details can be seen in appendix A.



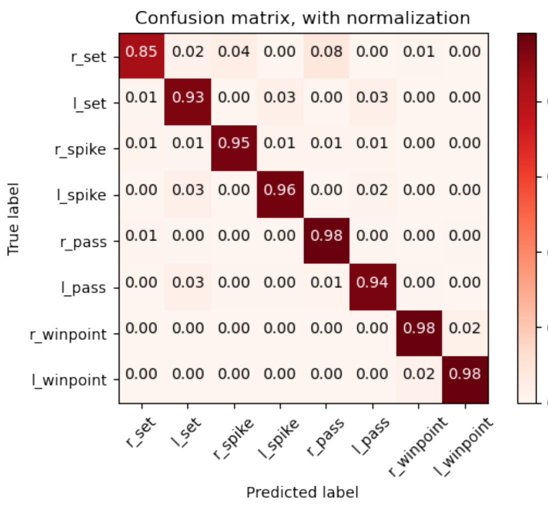
(a)



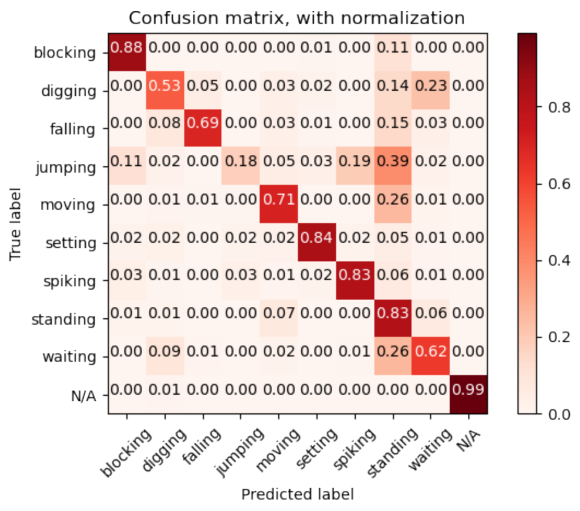
(b)

Figure 4.12: (a) confusion matrix for GAR with the COMPOSER ($accuracy = 95.0\%$) and (b) confusion matrix for HAR with the COMPOSER ($accuracy = 71.4\%$).

The above results are obtained while inputting the ball's annotated position to the model. In our end-to-end solution the ball annotations are not reliable, as mentioned previously in this chapter. Therefore, if we're using the COMPOSER model, we must use it without the ball annotations, which leads to worse results, but still on par with the other successfully tested models. The resulting confusion matrices of the training without the annotated ball position can be seen in figure 4.13



(a)



(b)

Figure 4.13: (a) confusion matrix for GAR with the COMPOSER ($accuracy = 94.62\%$) and (b) confusion matrix for HAR with the COMPOSER ($accuracy = 71.0\%$). Both without ball annotations and obtained with the same above described training conditions.

The confusion matrices are very similar to the ones obtained with the DECOMPL model, therefore, the same analysis that was previously made can also be applied to these matrices, with the exception residing in the fact that the COMPOSER's loss function depends on each scale's prediction.

4.2.6 Changes to the architecture of COMPOSER

Due to its rather low accuracy in the HAR task, some alterations were done to the COMPOSER model in an attempt to improve this accuracy, that can only achieve 71.4%. It's also observed that the ball's position and trajectory does not have a great effect on the model's accuracy, as without this data, the model's accuracy is 71.0%. The thought process is that the ball's position and trajectory should have a greater impact in this action classification, as for example, the trajectory of the ball in a spiking action should never be confused with that of a digging action, even if the player's detected keypoints are noisy.

To achieve this, we introduce a new scale before the group scale that represents the interactions between each player and the ball. This scale is parallel to the third scale, that represents persons interactions' with each other, and its output is used jointly with the second scale's output to predict the individual person action, as pictured in 4.14.

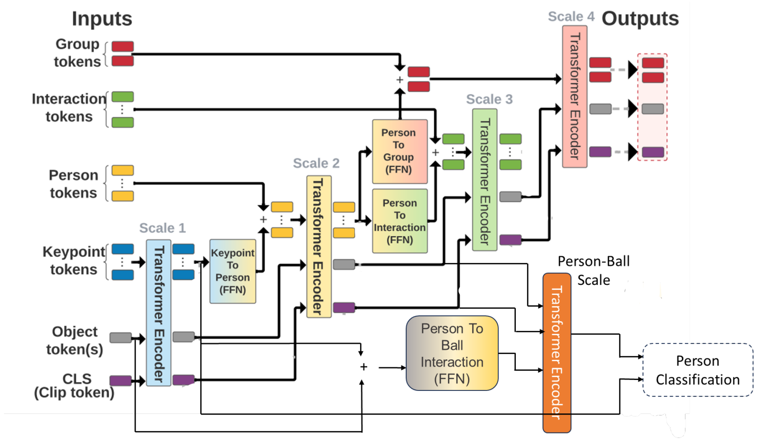
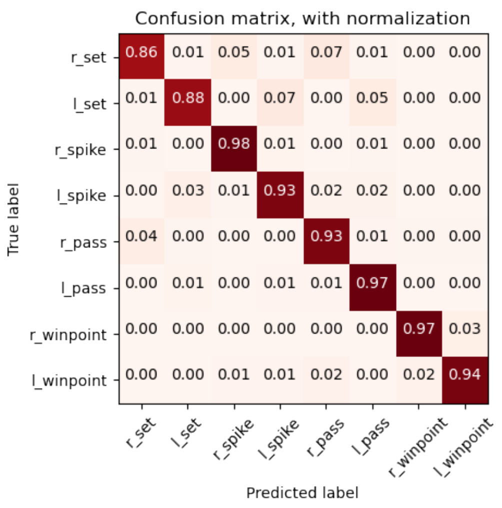
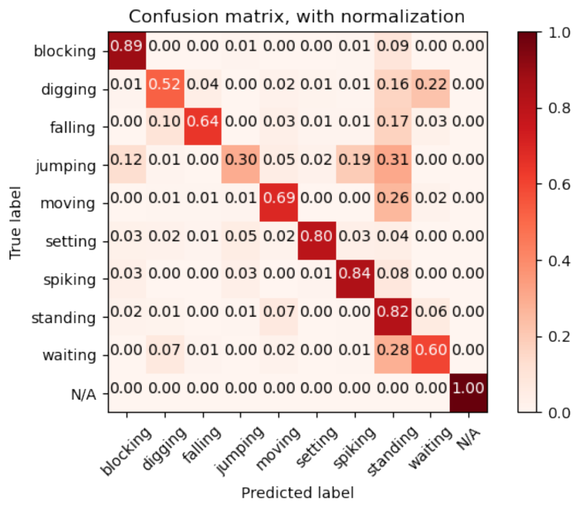


Figure 4.14: Architecture of the COMPOSER with the added person-ball interactions dimension.

This new layer, however, does not change the results of the model, which leads us to the conclusion that its addition is redundant and adds no new relevant information to the model. The results for the same hyperparameters can be seen in figure 4.15.



(a)



(b)

Figure 4.15: Results with the additional ball-person interaction layer. (a) confusion matrix for GAR ($accuracy = 93.25\%$) and (b) confusion matrix for HAR ($accuracy = 71.00\%$).

4.3 Results on Reannotations

As we want a model that performs well in reality and not just the best performing model in GAR benchmarks, it's mandatory that both the training labels and the test labels are correct. Therefore, we train the three best performing models that we could run (COMPOSER, DIN and DECOMPL) with the corrected annotations presented in chapter 2.

The following results are all obtained from training with the same hyperparameters as described in 4.2.

4.3.1 DIN

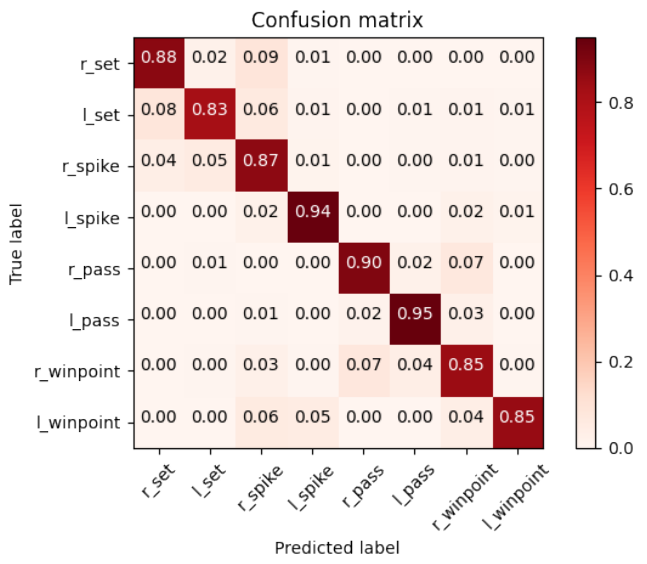
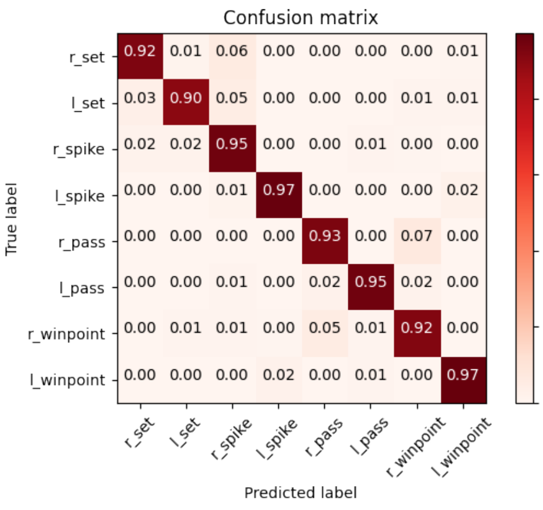


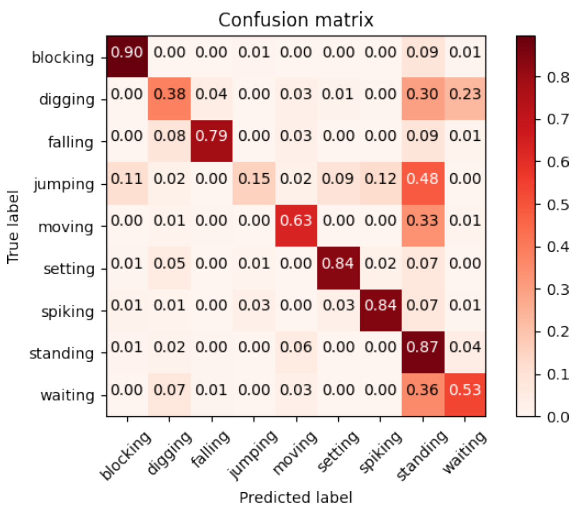
Figure 4.16: DIN confusion matrix for GAR on the corrected annotations ($accuracy = 82.37\%$).

4.3.2 DECOMPL



True \ Predicted	r_set	l_set	r_spike	l_spike	r_pass	l_pass	r_winpoint	l_winpoint
r_set	0.92	0.01	0.06	0.00	0.00	0.00	0.00	0.01
l_set	0.03	0.90	0.05	0.00	0.00	0.00	0.01	0.01
r_spike	0.02	0.02	0.95	0.00	0.00	0.01	0.00	0.00
l_spike	0.00	0.00	0.01	0.97	0.00	0.00	0.00	0.02
r_pass	0.00	0.00	0.00	0.00	0.93	0.00	0.07	0.00
l_pass	0.00	0.00	0.01	0.00	0.02	0.95	0.02	0.00
r_winpoint	0.00	0.01	0.01	0.00	0.05	0.01	0.92	0.00
l_winpoint	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.97

(a)

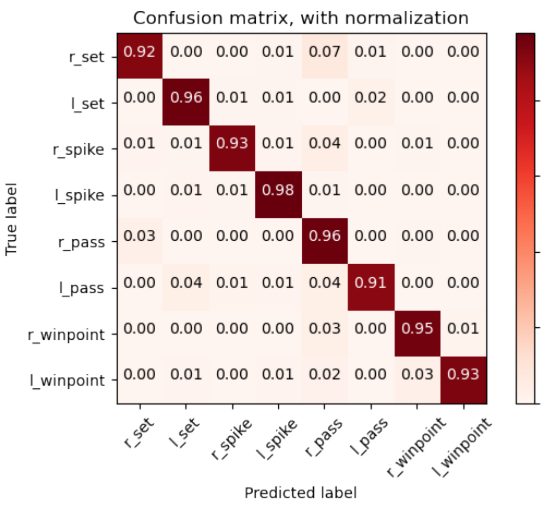


True \ Predicted	blocking	digging	falling	jumping	moving	setting	spiking	standing	waiting
blocking	0.90	0.00	0.00	0.01	0.00	0.00	0.00	0.09	0.01
digging	0.00	0.38	0.04	0.00	0.03	0.01	0.00	0.30	0.23
falling	0.00	0.08	0.79	0.00	0.03	0.00	0.00	0.09	0.01
jumping	0.11	0.02	0.00	0.15	0.02	0.09	0.12	0.48	0.00
moving	0.00	0.01	0.00	0.00	0.63	0.00	0.00	0.33	0.01
setting	0.01	0.05	0.00	0.01	0.00	0.84	0.02	0.07	0.00
spiking	0.01	0.01	0.00	0.03	0.00	0.03	0.84	0.07	0.01
standing	0.01	0.02	0.00	0.00	0.06	0.00	0.00	0.87	0.04
waiting	0.00	0.07	0.01	0.00	0.03	0.00	0.00	0.36	0.53

(b)

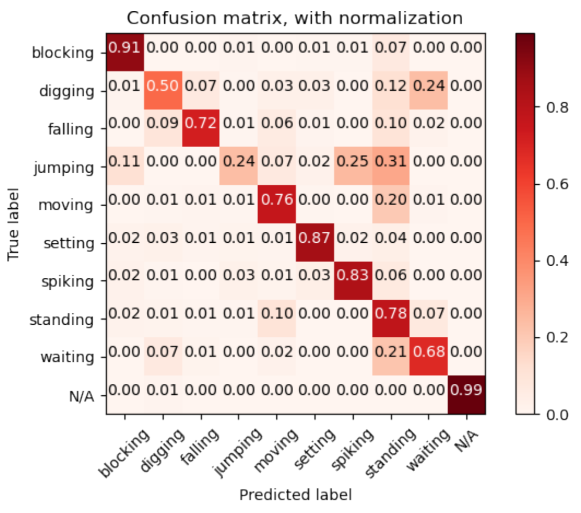
Figure 4.17: DECOMPL results with training with the corrected annotations. (a) confusion matrix for GAR ($accuracy = 93.87\%$)and (b) confusion matrix for HAR ($accuracy = 65.88\%$).

4.3.3 COMPOSER



True \ Predicted	r_set	l_set	r_spike	l_spike	r_pass	l_pass	r_winpoint	l_winpoint
r_set	0.92	0.00	0.00	0.01	0.07	0.01	0.00	0.00
l_set	0.00	0.96	0.01	0.01	0.00	0.02	0.00	0.00
r_spike	0.01	0.01	0.93	0.01	0.04	0.00	0.01	0.00
l_spike	0.00	0.01	0.01	0.98	0.01	0.00	0.00	0.00
r_pass	0.03	0.00	0.00	0.00	0.96	0.00	0.00	0.00
l_pass	0.00	0.04	0.01	0.01	0.04	0.91	0.00	0.00
r_winpoint	0.00	0.00	0.00	0.00	0.03	0.00	0.95	0.01
l_winpoint	0.00	0.01	0.00	0.01	0.02	0.00	0.03	0.93

(a)



True \ Predicted	blocking	digging	falling	jumping	moving	setting	spiking	standing	waiting	N/A
blocking	0.91	0.00	0.00	0.01	0.00	0.01	0.01	0.07	0.00	0.00
digging	0.01	0.50	0.07	0.00	0.03	0.03	0.00	0.12	0.24	0.00
falling	0.00	0.09	0.72	0.01	0.06	0.01	0.00	0.10	0.02	0.00
jumping	0.11	0.00	0.00	0.24	0.07	0.02	0.25	0.31	0.00	0.00
moving	0.00	0.01	0.01	0.01	0.76	0.00	0.00	0.20	0.01	0.00
setting	0.02	0.03	0.01	0.01	0.01	0.87	0.02	0.04	0.00	0.00
spiking	0.02	0.01	0.00	0.03	0.01	0.03	0.83	0.06	0.00	0.00
standing	0.02	0.01	0.01	0.01	0.10	0.00	0.00	0.78	0.07	0.00
waiting	0.00	0.07	0.01	0.00	0.02	0.00	0.00	0.21	0.68	0.00
N/A	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99

(b)

Figure 4.18: COMPOSER results with training with the corrected annotations. (a) confusion matrix for GAR ($accuracy = 94.25\%$)and (b) confusion matrix for HAR ($accuracy = 72.8\%$).

4.3.4 Comparison

For all the three models, DIN, DECOMPL and COMPOSER we see an increase in performance with the corrected annotations, but we still have the same standards, with COMPOSER achieving the highest accuracy, DECOMPL the second highest and DIN the lowest.

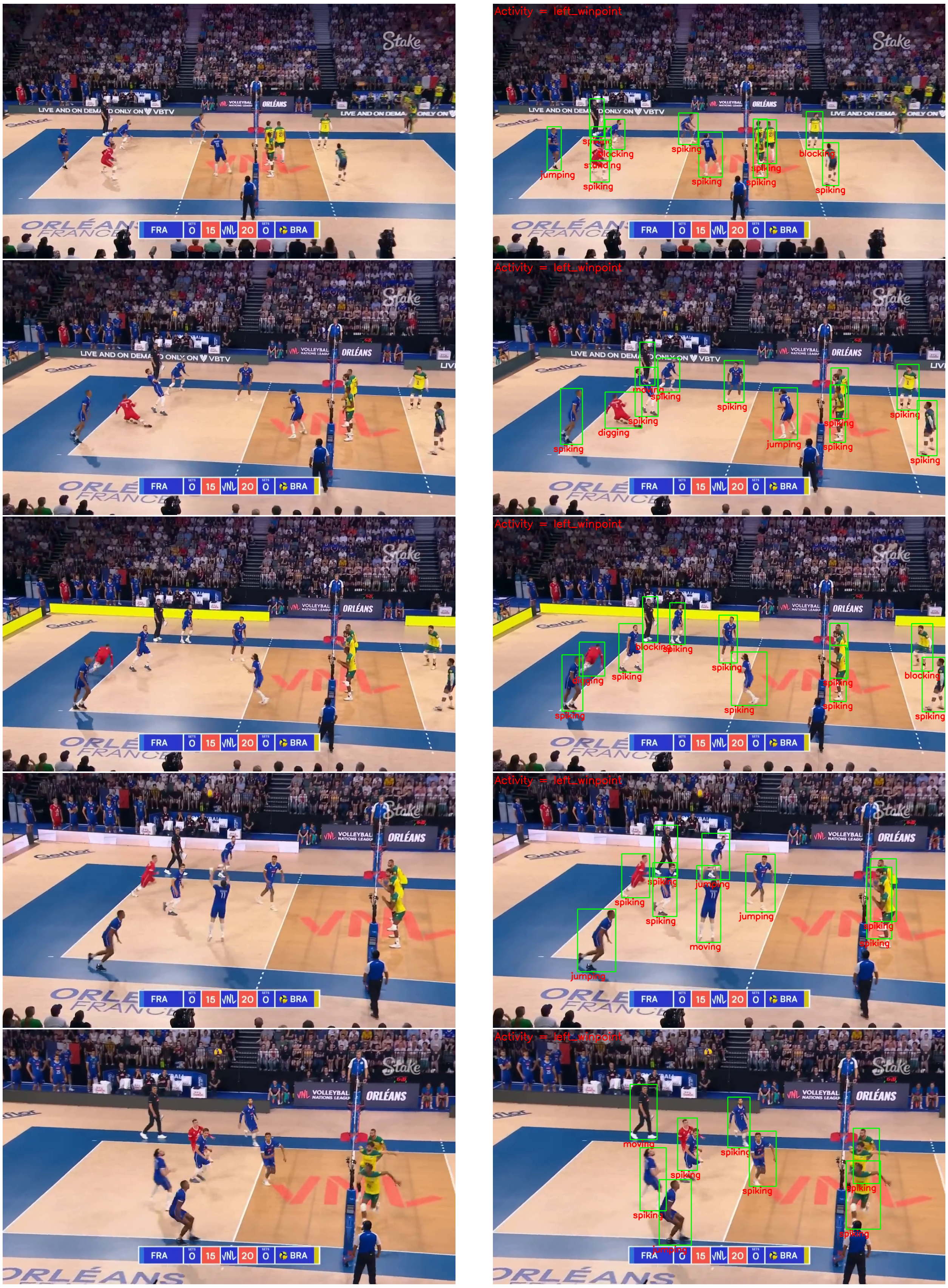
In possible future work that puts more focus on HAR, it's also important to reannotate the VD action labels, that are also incorrectly defined, because as we can see with the GAR labels it's important that the labels are correctly annotated.

4.4 Results on Test Video

Since the test video is not annotated as it serves the purpose to test how viable the inference is, there are no metrics available that were measured on this data. For that purpose, we already used the VD train-test split.

We can, however, visualize the results. Since HAR is already very inaccurate during validation, it's expected that the results are even worse in the new video inference, where the frames are different and, frequently, are not centered around the exact frame where the activity takes place. From these examples we should see how the final video output is, and while practical applications might not have the detections printed on the frames, this example still serves as a visualization of what the model on the detections on a different volleyball video is capable of.

4.4.1 DECOMPL


Activity = left_winpoint
jumping
spiking
blocking
standing
spiking
spiking
spiking
spiking
spiking
blocking
spiking
Activity = left_winpoint
moving
spiking
spiking
digging
spiking
spiking
jumping
spiking
spiking
spiking
spiking
Activity = left_winpoint
blocking
spiking
spiking
spiking
spiking
spiking
spiking
spiking
spiking
blocking
spiking
Activity = left_winpoint
spiking
spiking
jumping
spiking
moving
jumping
spiking
spiking
jumping
Activity = left_winpoint
moving
spiking
spiking
spiking
spiking
jumping
spiking
spiking
FRA
BRA
VNL
ORLÉANS
Stake

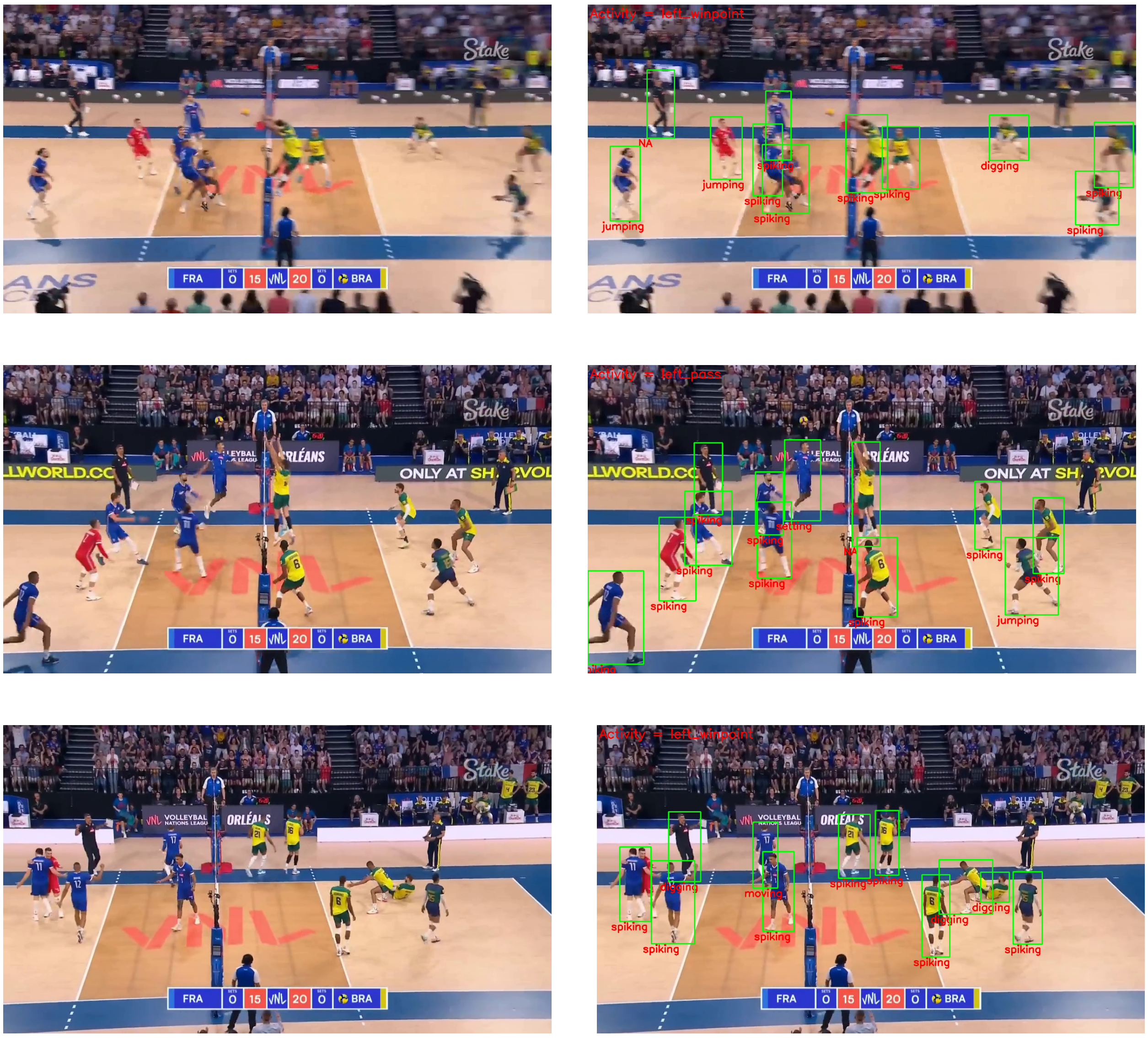


Figure 4.20: DECOMPL inference. We show the first 6 video segments (of 20 frames) and the last two. The frames without detections are displayed on the left and the corresponding frame with the DECOMPL predictions on the right.

Table 4.2: DECOMPL predictions of the images 4.20, in corresponding order. (l-win = left-winpoint)

Label	r-pass	l-pass	l-pass	l-set	l-spike	r-spike	l-spike	l-winpoint
Predicted	*l-win*	*l-win*	*l-win*	*l-win*	*l-win*	*l-win*	*l-pass*	l-win

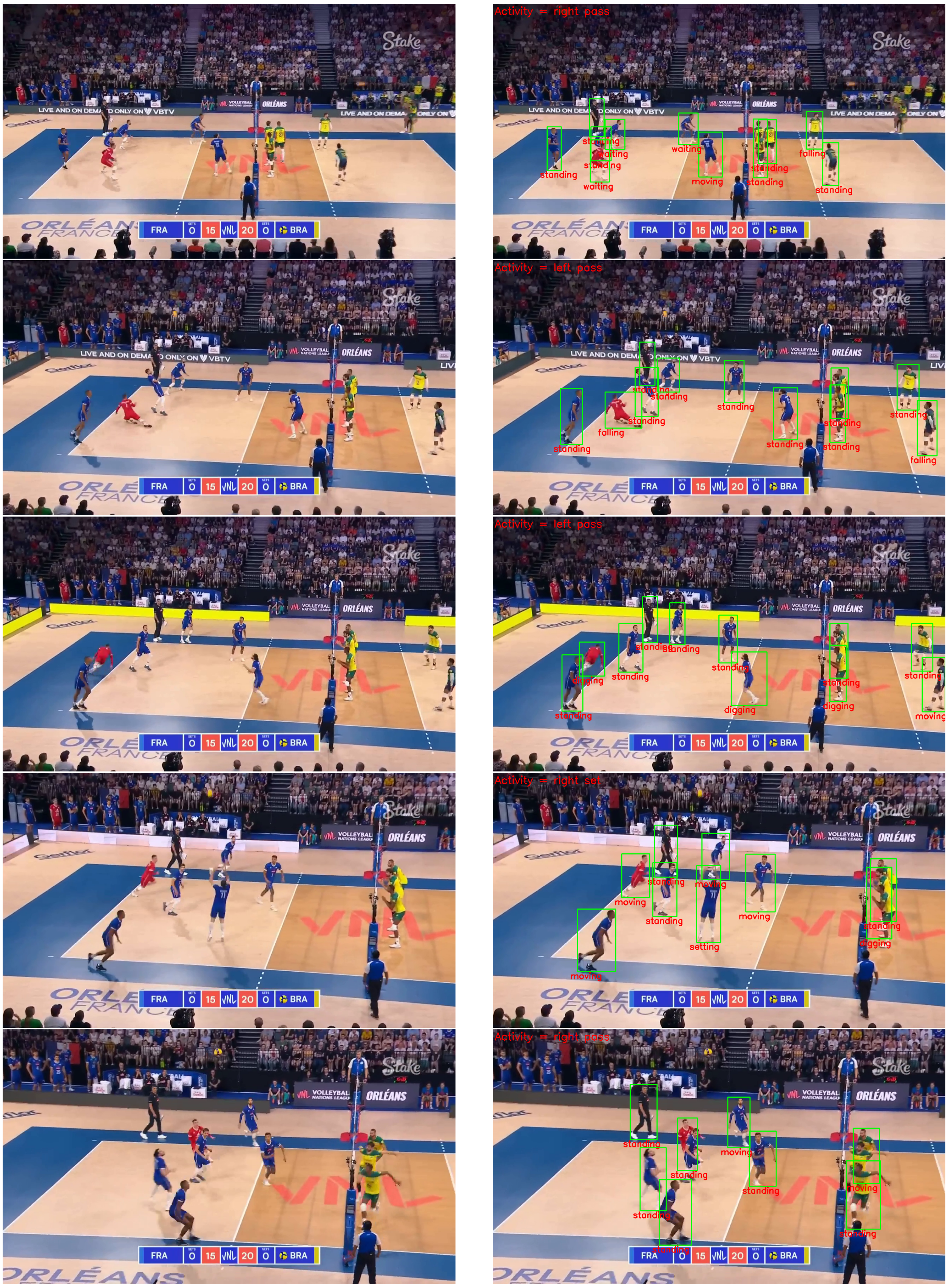
In the test dataset, the DECOMPL's accuracy is 6.67%, as the model predicts left winpoint almost consistently, as if the weights were uninitialized. Some decline in performance was expected the new data inference, however, such a drastic decrease of 84.95% over the original 91.62% was unforeseen. This could have happened for two reasons:

- Scene biases - Since the DECOMPL is an RGB based model and the test dataset contains a different scene from the one of the VD, it could be possible that the model didn't generalize well enough. However, due to data augmentation techniques, that resize, change the colors and crop the image, this should not be an issue.

- Missing players - The test video camera point of view focuses more on the playing side, especially when the left side is playing, and ends up not recording a few players of the opposing team for a longer time frame; as opposed to the VD where the whole field is exposed to the camera at all times except for winpoint situations. Based on the actor dropout data augmentation technique, we replace the bounding boxes of missing detections with null vectors, which could be causing this poor performance of DECOMPL, and thus this is the most likely reason to be causing the issue.

We exclude the fact of the central frame not containing the actual instant in which the activity takes place, because the DECOMPL prediction is made on a single image at a time, and then is averaged across all the frames in one sequence to reach the final prediction that is output. In fact, for this reason, the DECOMPL could be expected to be the best performing model in inference conditions. For future work, it might be possible to study the alternative methods of handling missing players, to find the reason for the DECOMPL underperformance.

4.4.2 COMPOSER



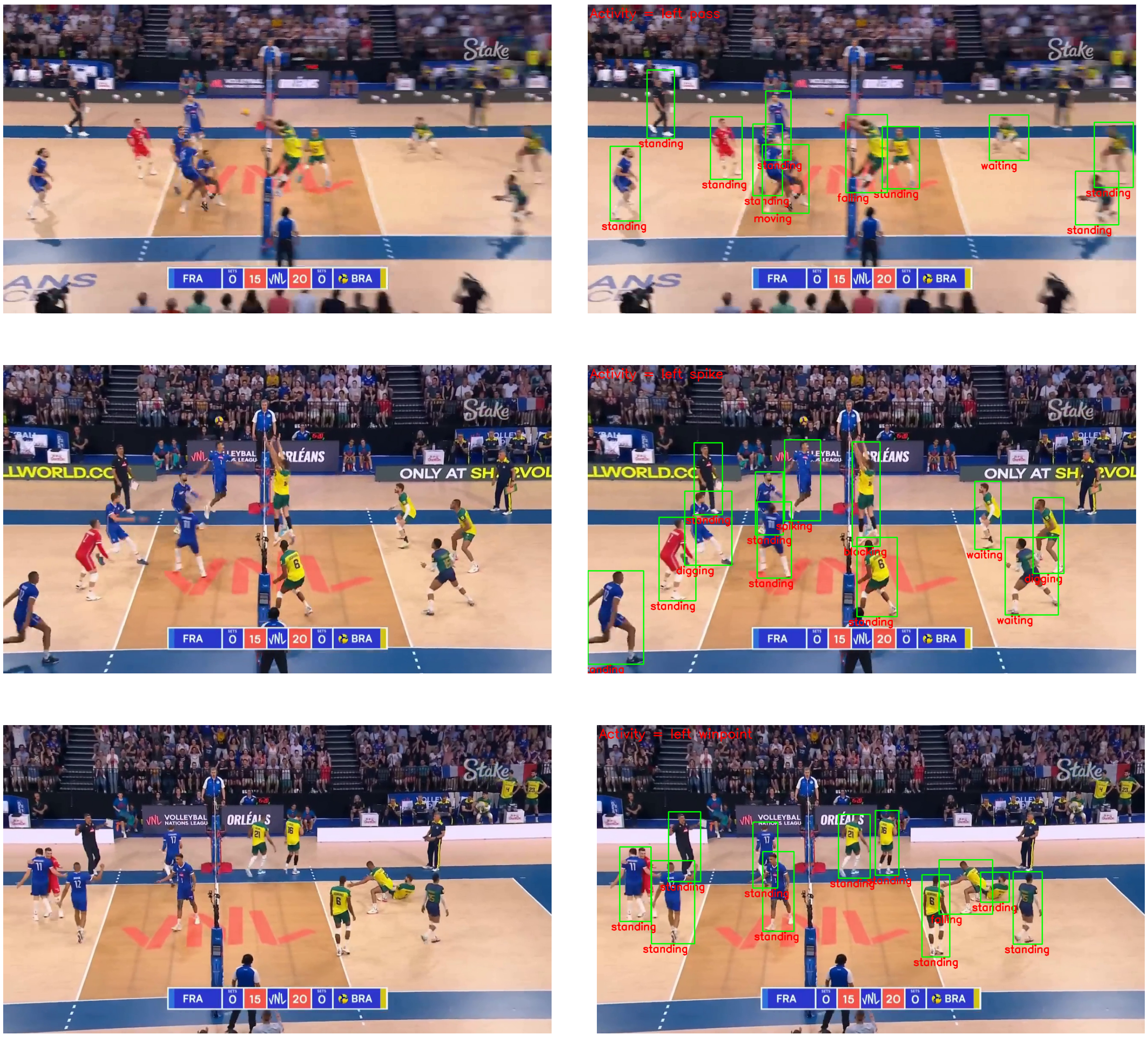


Figure 4.22: COMPOSER inference. We show the first 6 video segments (of 20 frames) and the last two. The frames without detections are displayed on the left and the corresponding frame with the COMPOSER predictions on the right.

Table 4.3: COMPOSER predictions of the images 4.22, in corresponding order.

Label	r-pass	l-pass	l-pass	l-set	l-spike	r-spike	l-spike	l-winpoint
Predicted	r-pass	l-pass	l-pass	*r-set*	*r-pass*	*l-pass*	l-spike	l-winpoint

From table 4.3 we see that from the sample analyzed, only 5 in 8 GAR predictions are correct. Of the three wrong ones, the first (right set instead of left set) misses the side, the second one (right pass instead of left spike) misses both the side and the group activity: it's one of the cases where the central frame is not quite the one where the activity occurs; and finally the third miss is also completely off the mark, having a prediction of left pass in place of a right spike, which should be a very rare case, given the model's accuracy, that's greater than 90%.

Even though the remaining results aren't presented, COMPOSER made 19 right predictions and 11 wrong predictions in the unannotated dataset, which can be translated in 63.33% accuracy. It was expected that the accuracy would be lower in this the test data, however, a decrease in almost 30% over the validation accuracy of 95.00% might make the method not accurate enough to be employed in real application scenarios. Nevertheless, we must also keep in mind that this 63.33% accuracy is obtained in only 30 segments of video (around 90 seconds), which is a small number to truly evaluate the accuracy of the model: in a full video of a volleyball game that could last up to 90 minutes, the results could agree with this accuracy, but they could also be better or even worse.

5 Conclusion

5.1 Final Dissertation remarks

This Dissertation ends up evaluating three aspects that are relevant for automatic video analysis through DL techniques: object detection, object tracking and action and activity recognition, which is the main goal for the video analysis. We implement SOTA models for each of these subjects and can conclude the following:

- OD can be performed by YOLO models with acceptable results; in fact, the detections are never perfect, but the tracking that is applied afterwards improves their accuracy. YOLOv5-x is the YOLO model that could achieve the best results, with an F1-Score of 75%
- Tracking is very reliable, as long as the previous object detections are correct.
- GAR and HAR are dependent on the detections and tracking performed. Although the models can perform very well on correctly annotated data achieving accuracies greater than 90% for the DECOMPL and COMPOSER models, when the frames are not centered around the action and the camera angle shifts to one team's side, the results are much worse than on validation.

With the proposed architecture, the COMPOSER is the only model that could possibly be used in automatic analysis and highlight reel generation. However, COMPOSER requires detecting keypoints beforehand, which takes a long time (4 minutes for 639 frames). For this reason, it's not possible to use it for real-time analysis. Even when using on already recorded video, the accuracy of the COMPOSER is only 63%, which might not be enough to provide satisfying results. This accuracy, however, depends on the nature of the video: logically, if the video is more similar to the training data, the results will also be better.

5.2 Future work

For future work, there are many improvements that can be made to all the DL models used. The detector employed can be improved, and it's possible that as DL architectures evolve, better detectors will naturally be developed, and thus improve the detection quality.

When it comes to GAR and HAR, some things could be considered to improve the accuracy of the models in new, different data:

- Study ways to overcome the problem of actors leaving the scene, which seems to be affecting the way the models generalize to unforeseen data.
- Expand the VD in order to train in a more varied dataset. The Volleyball Dataset contains videos from only a couple of different events, that were recorded under the same conditions and thus the videos are very similar between them. Even with the employed data augmentation techniques like cropping, flipping and color augmentation, the training could benefit from having a larger, more diverse dataset.
- Training on the VD while changing the central frame. This could be achieved by implementing a form of data augmentation where the central frame does not represent the exact frame where the activity takes place, which would be offset by some frames. This data augmentation could have the model generalize better to instances that were segmented automatically, like we have when applying GAR and HAR inference of new unannotated videos.
- For HAR, the VD annotations should be reviewed in the first place. Although it can be a challenging task, as during one activity one player might perform multiple actions, e.g. moving and jumping, we have seen that with corrected annotations, the performance of the models can only be improved. Then, short duration action recognition models could be tested, or the loss functions of the tested GAR solutions could be changed in order to better perform HAR alongside GAR.

Different sampling of the frames could also be looked into. We uniformly sample the frames 20 by 20, but there are alternatives: the models could be trained to run on fewer frames, and the predictions could be made based on the average predictions for multiple sequential samples. For instance, instead of 20 if we had 5 frames per sample, we would have 4 samples for each sample that we currently have. The model would make 4 predictions for each prediction that we have, and the predictions could be averaged to get to the activity label. This technique could create better limits to the group activity, as sometimes an activity can happen so quickly that it's skipped due to the temporal size of each sample.

Bibliography

[1] Fei Wu et al. "A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications". In: *arXiv preprint arXiv:2206.01038* (2022).

[2] Honglu Zhou et al. "COMPOSER: Compositional Learning of Group Activity in Videos". In: *arXiv preprint arXiv:2112.05892* (2021).

[3] Haodong Duan et al. "Revisiting skeleton-based action recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2969–2978.

[4] Mostafa S. Ibrahim et al. "Hierarchical Deep Temporal Models for Group Activity Recognition." In: 2016.

[5] Berker Demirel and Huseyin Ozkan. "DECOMPL: Decompositional Learning with Attention Pooling for Group Activity Recognition from a Single Volleyball Image". In: *arXiv preprint arXiv:2303.06439* (2023).

[6] Mauricio Perez, Jun Liu, and Alex C Kot. "Skeleton-based relational reasoning for group activity analysis". In: *Pattern Recognition* 122 (2022), p. 108360.

[7] Kohei Sendo and Norimichi Ukita. "Heatmapping of people involved in group activities". In: *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE. 2019, pp. 1–6.

[8] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[9] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[10] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[11] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[12] Jianchao Wu et al. "Learning actor relation graphs for group activity recognition". In: *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*. 2019, pp. 9964–9974.

[13] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[14] Anurag Arnab et al. "Vivit: A video vision transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6836–6846.

[15] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[16] Hangjie Yuan, Dong Ni, and Mang Wang. "Spatio-temporal dynamic inference network for group activity recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 7476–7485.

[17] Jiquan Ngiam et al. "Scene transformer: A unified architecture for predicting multiple agent trajectories". In: *arXiv preprint arXiv:2106.08417* (2021).

[18] Yuxin Wu et al. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[19] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[20] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[21] Yifu Zhang et al. "Bytetrack: Multi-object tracking by associating every detection box". In: *European Conference on Computer Vision*. Springer. 2022, pp. 1–21.

[22] Fangao Zeng et al. "Motr: End-to-end multiple-object tracking with transformer". In: *European Conference on Computer Vision*. Springer. 2022, pp. 659–675.

[23] Bowen Cheng et al. "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 5386–5395.

[24] Masato Tamura, Rahul Vishwakarma, and Ravigopal Vennelakanti. "Hunting Group Clues with Transformers for Social Group Activity Recognition". In: *arXiv preprint arXiv:2207.05254* (2022).

[25] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[26] Meng Joo Er et al. "Attention pooling-based convolutional neural network for sentence modelling". In: *Information Sciences* 373 (2016), pp. 388–403.

[27] Shuaicheng Li et al. “Groupformer: Group activity recognition with clustered spatial-temporal transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 13668–13677.

[28] Will Kay et al. “The kinetics human action video dataset”. In: *arXiv preprint arXiv:1705.06950* (2017).

[29] AJ Piergiovanni, Weicheng Kuo, and Anelia Angelova. “Rethinking video vits: Sparse video tubes for joint image and video learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2023, pp. 2214–2224.

[30] Chuyi Li et al. “Yolov6 v3. 0: A full-scale reloading”. In: *arXiv preprint arXiv:2301.05586* (2023).

[31] Zheng Ge et al. “Yolox: Exceeding yolo series in 2021”. In: *arXiv preprint arXiv:2107.08430* (2021).

[32] Peize Sun et al. “Dancetrack: Multi-object tracking in uniform appearance and diverse motion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 20993–21002.

[33] Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. “Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2023, pp. 22056–22065.

Appendix A

YOLO results

In this appendix multiple results output by the YOLO training and validation process are presented to compare different results with different number of epochs, image sizes and weights. The learning rate is initialized at 0.001, the momentum is 0.937 and the weight decay is 0.00005 for all runs.

A.1 YOLOv8 results

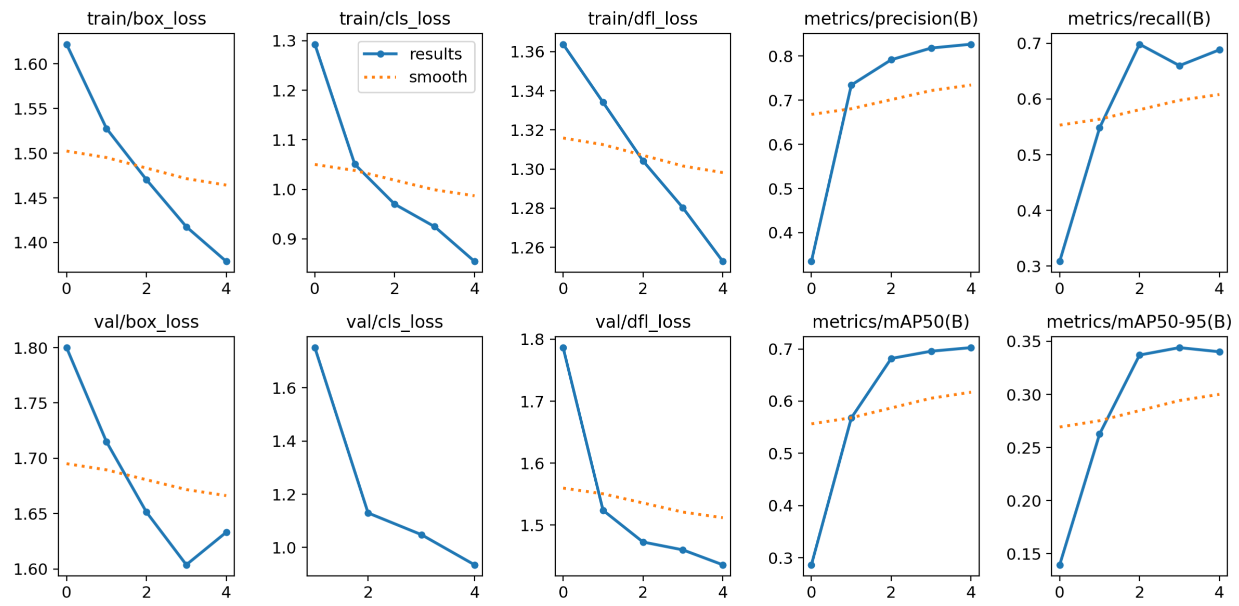


Figure A.1: Metrics for YOLOv8 with the yolov8n weights, image size 1280 over 5 epochs.

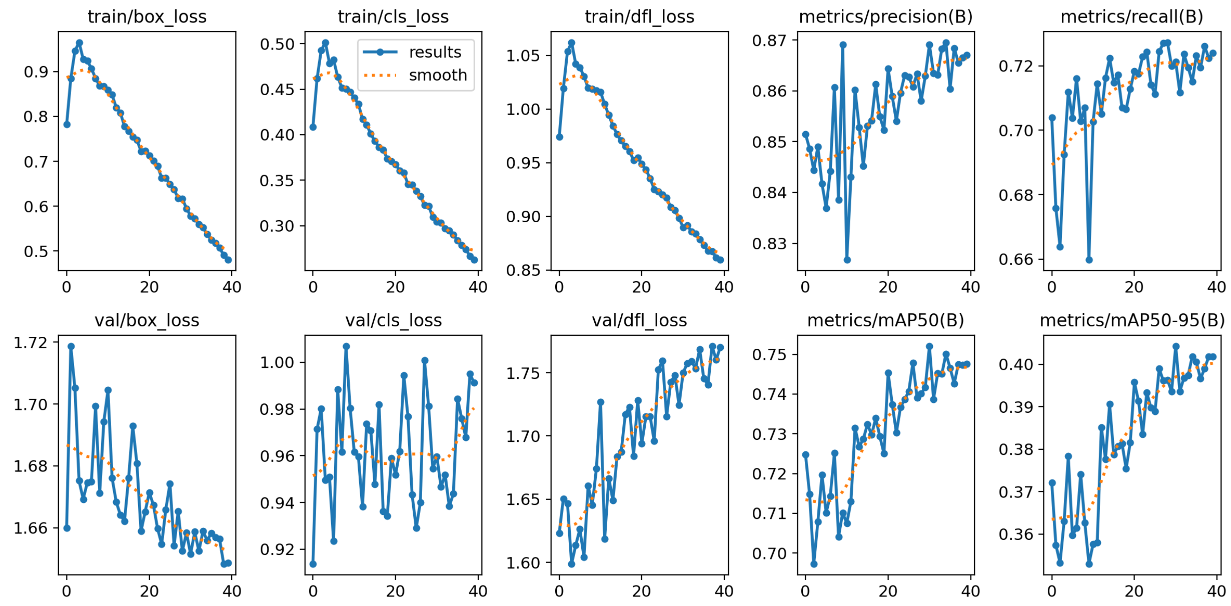


Figure A.2: Metrics for YOLOv8 with the yolov8n weights, image size 1280 over 40 epochs.

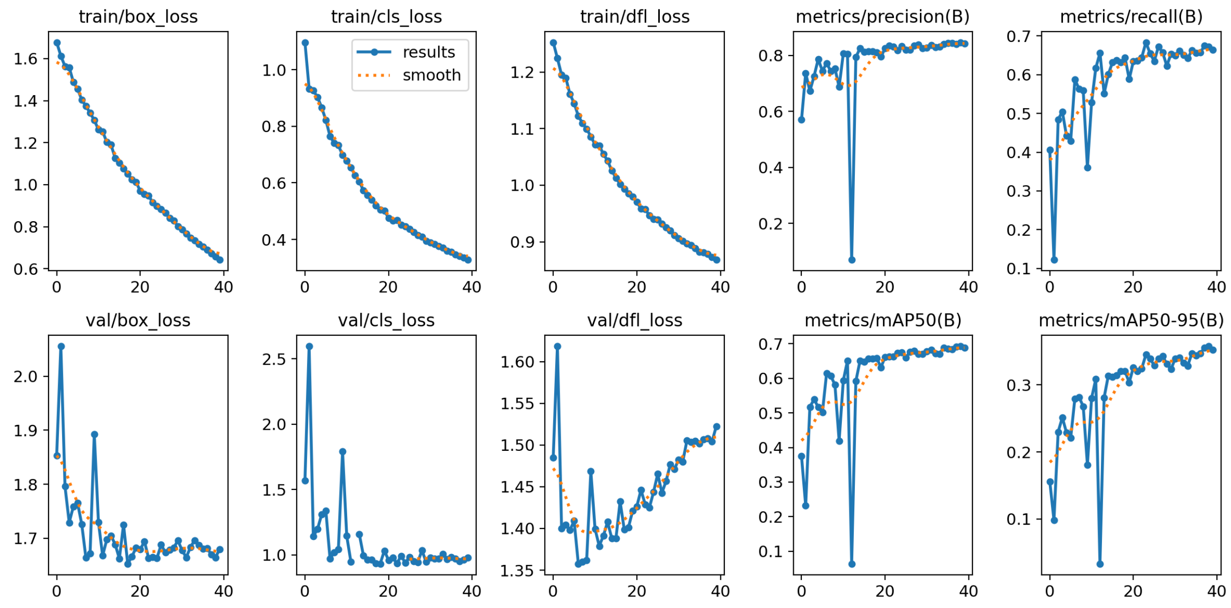


Figure A.3: Metrics for YOLOv8 with the yolov8x weights, image size 640 over 40 epochs.

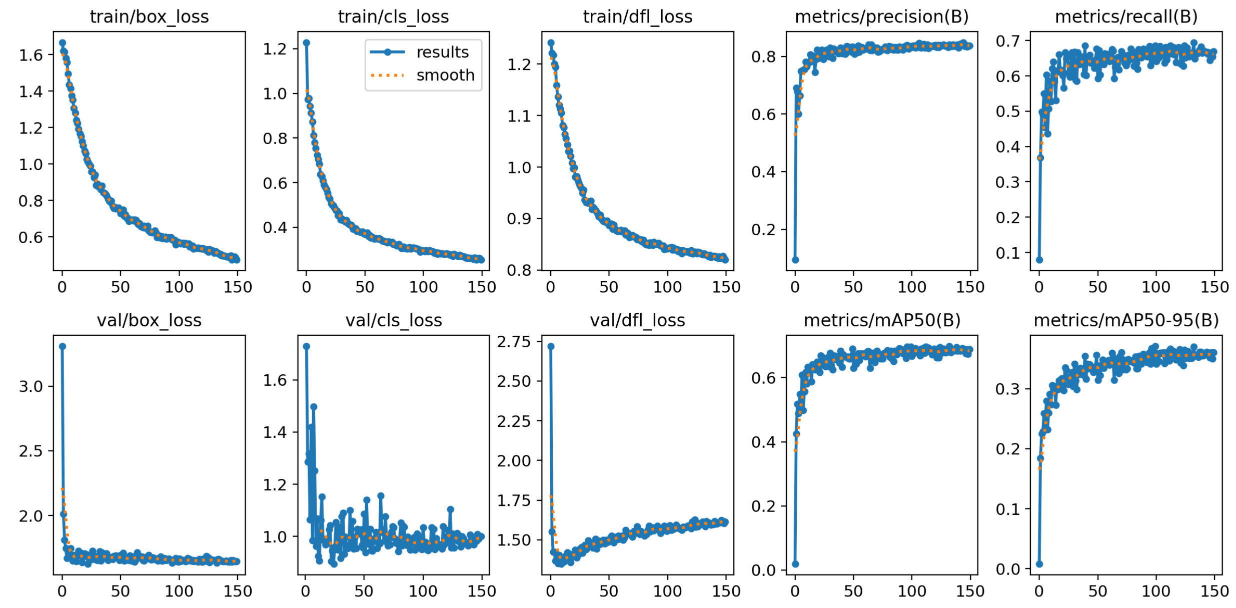


Figure A.4: Metrics for YOLOv8 with the yolov8x weights, image size 640 over 300 epochs.

A.2 YOLOv5 results

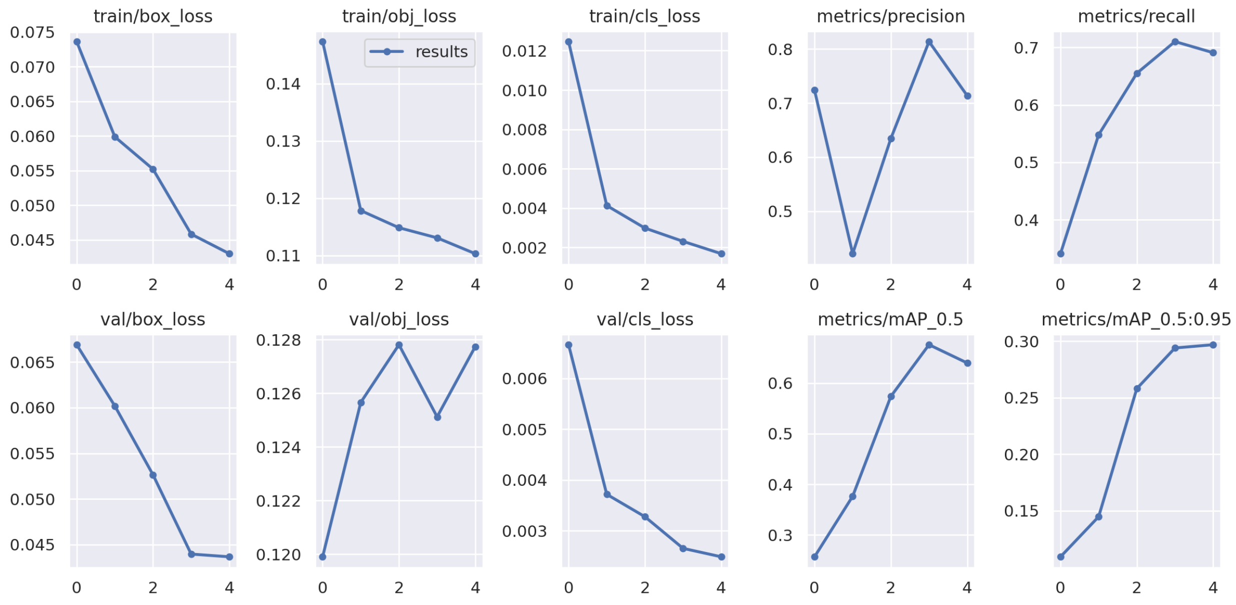


Figure A.5: Metrics for YOLOv5 with the yolov5x weights, image size 640 over 5 epochs.

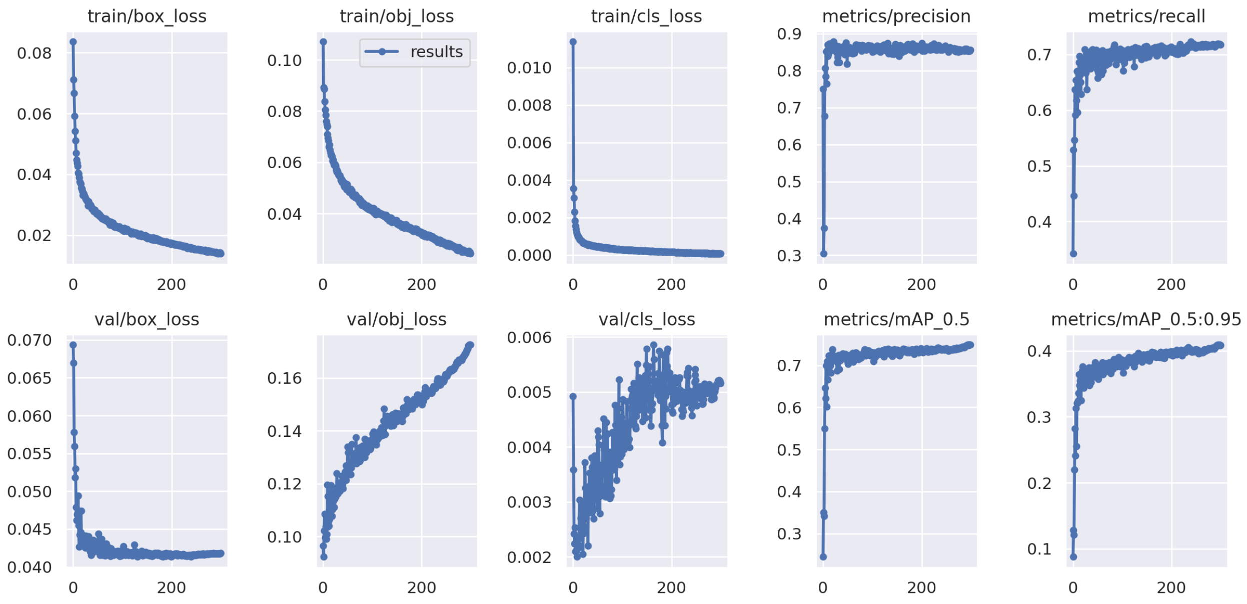
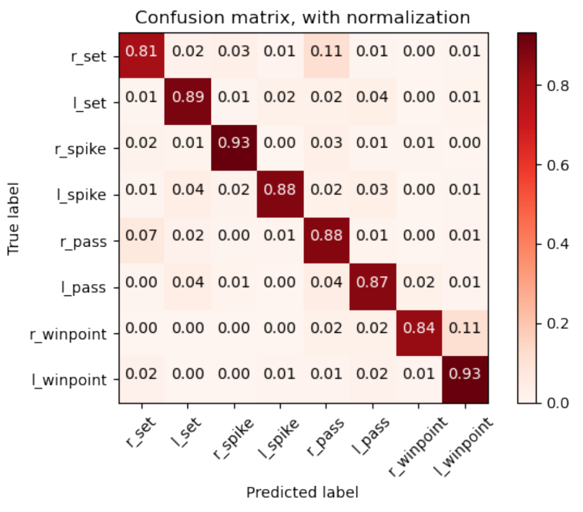


Figure A.6: Metrics for YOLOv5 with the yolov5x weights, image size 640 over 300 epochs (final model).

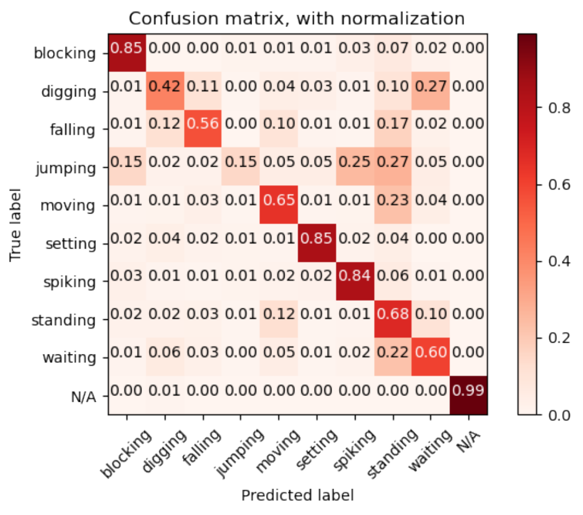
Appendix B

Different COMPOSER results

Results for diverse COMPOSER runs are presented. By default, the training is done on 10 480×853 frames, sampling 5 frames before the central frame and 4 frames after it and ball annotations are not used. The hyperparameters are the same as the ones described in 4.

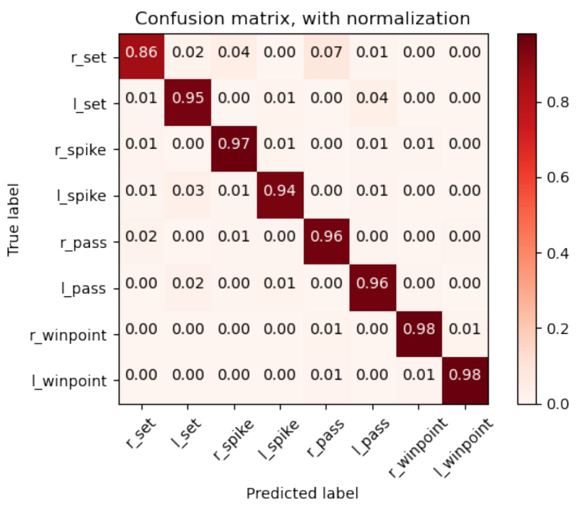


(a)

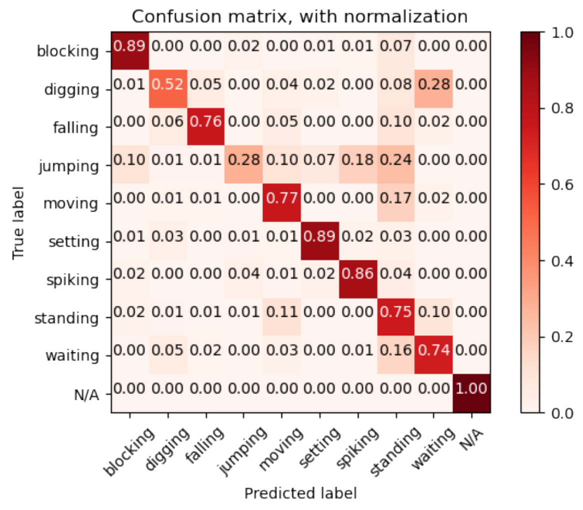


(b)

Figure B.1: COMPOSER results while training on 480×853 frames. (a) confusion matrix for GAR ($accuracy = 87.87\%$)and (b) confusion matrix for HAR ($accuracy = 66.00\%$).

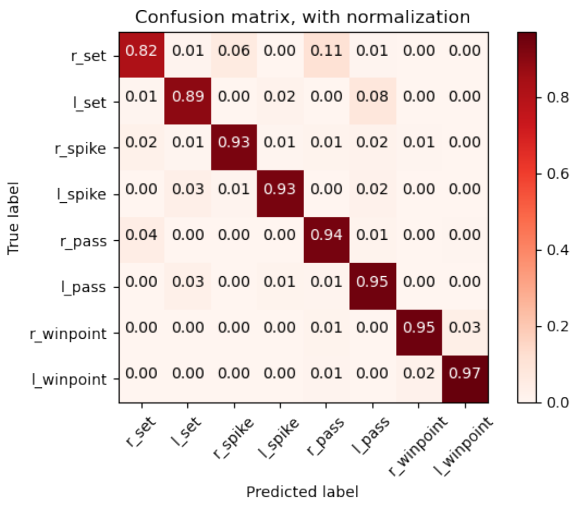


(a)

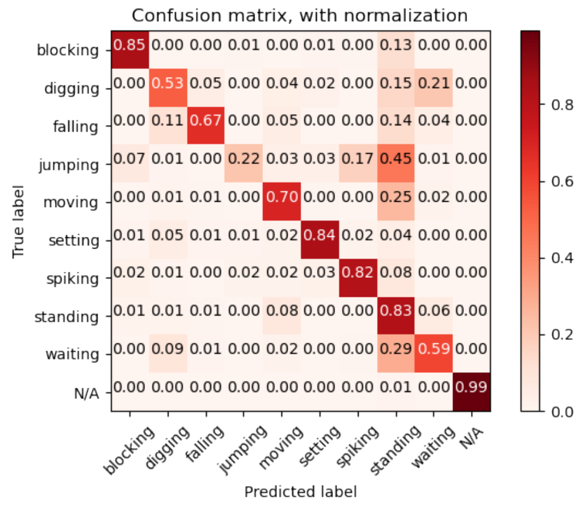


(b)

Figure B.2: Training on only 5 frames, with ball annotations. (a) confusion matrix for GAR ($accuracy = 95.00\%$)and (b) confusion matrix for HAR ($accuracy = 74.6\%$).

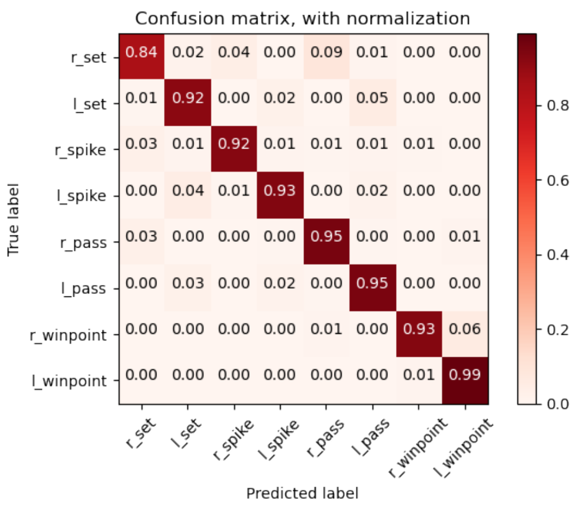


(a)

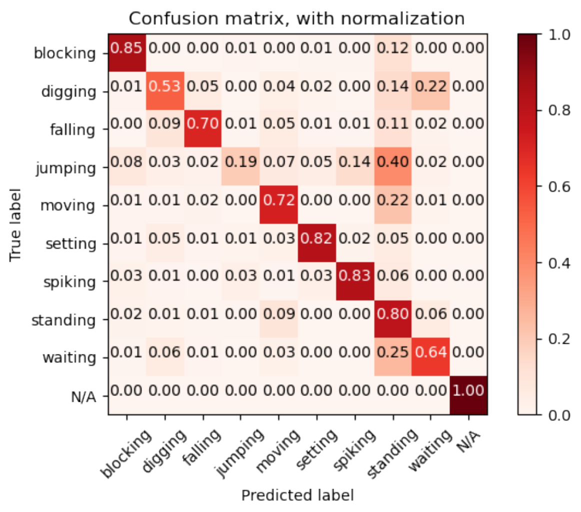


(b)

Figure B.3: Training on only 2 frames (central and the one before). (a) confusion matrix for GAR ($accuracy = 92.25\%$)and (b) confusion matrix for HAR ($accuracy = 70.40\%$).

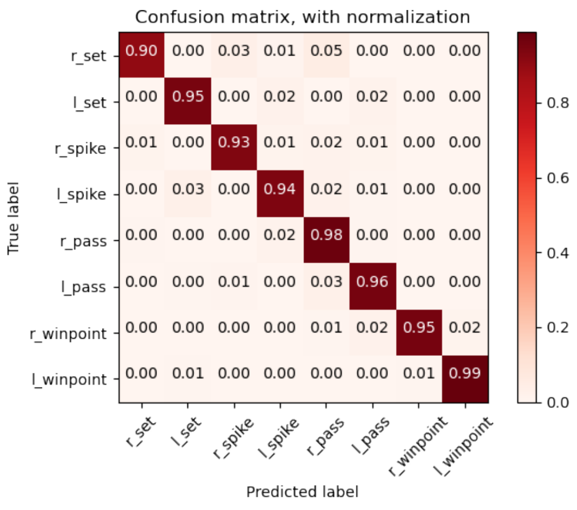


(a)

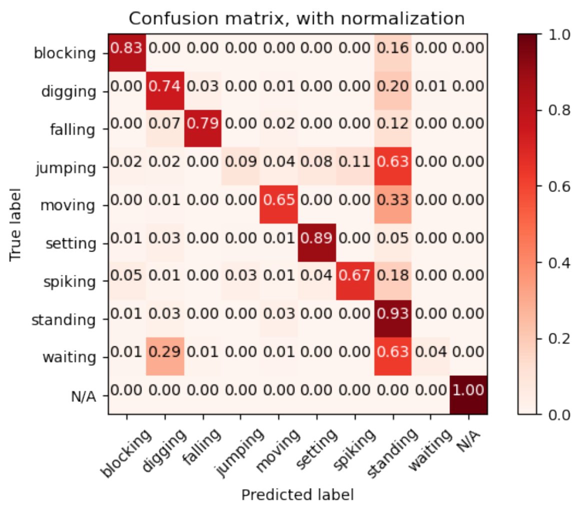


(b)

Figure B.4: Training on only 2 frames (central and the fourth before). (a) confusion matrix for GAR ($accuracy = 92.87\%$)and (b) confusion matrix for HAR ($accuracy = 71.7\%$).

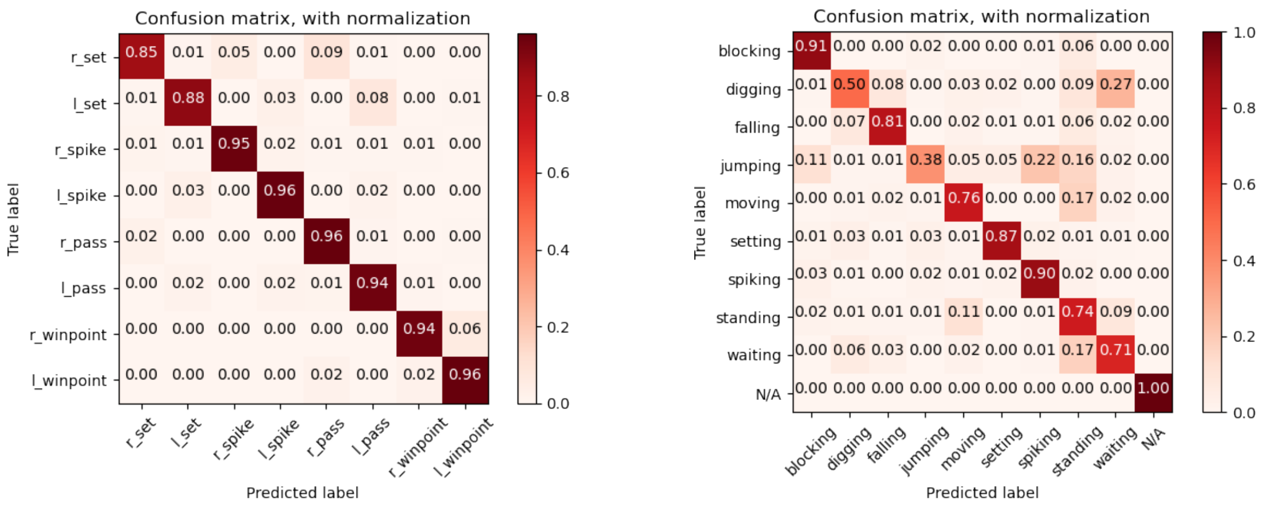


(a)



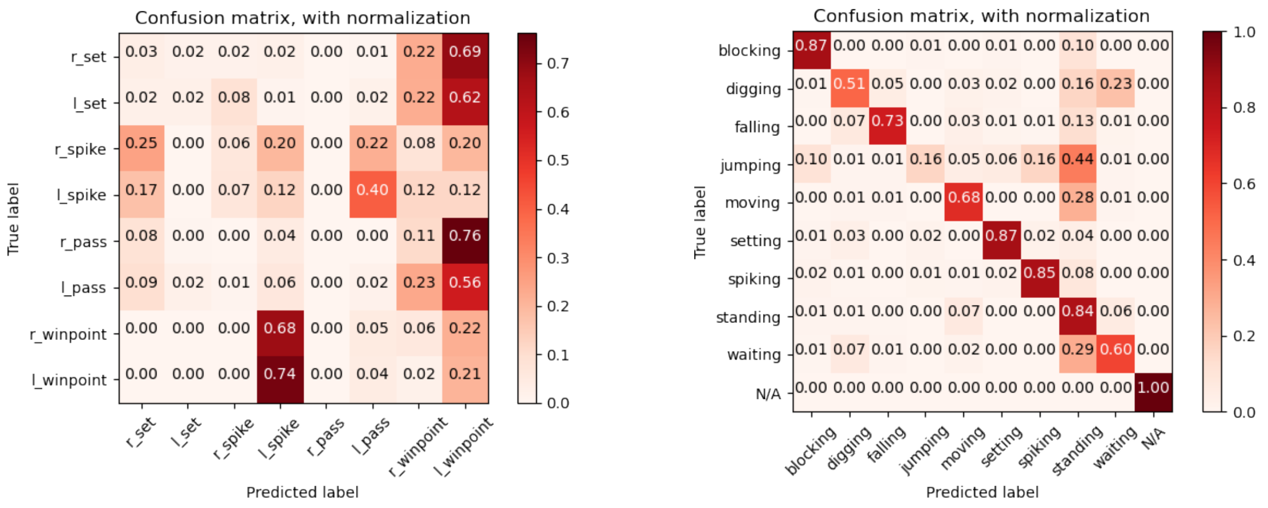
(b)

Figure B.5: Training on 10 frames, sampling every 2 frames. (a) confusion matrix for GAR ($accuracy = 95.00\%$)and (b) confusion matrix for HAR ($accuracy = 66.3\%$).



(a) (b)

Figure B.6: Changing the loss function to have the person loss with as much weight as the scale's loss combined (multiplying it by 6). (a) confusion matrix for GAR ($accuracy = 93.00\%$)and (b) confusion matrix for HAR ($accuracy = 75.8\%$).



(a) (b)

Figure B.7: Training only for action recognition (no loss calculated for group activity). (a) confusion matrix for GAR ($accuracy = 6.50\%$)and (b) confusion matrix for HAR ($accuracy = 71.1\%$).