UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Basketball Pose-based Action Recognition

Author:
Iryna ZAKHARCHENKO

Supervisor:
Orest VARGA

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Iryna ZAKHARCHENKO, declare that this thesis titled, "Basketball Pose-based Action Recognition" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Push yourself again and again. Don't give an inch until the final buzzer sounds."

Larry Bird

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Basketball Pose-based Action Recognition

by Iryna ZAKHARCHENKO

Abstract

Action detection on a team sport is a challenging task, while sports analysis is on-demand and in high interest. A great number of researchers try to make analysis automated. Despite enormous success in image classification using deep learning, action recognition in the video remains a difficult task, and at present no good solution exists in terms of accuracy and speed. The main challenge in action recognition is to design architecture that will capture both spatial and temporal information. In team sports action analysis, the serious challenges are that we have multiple players performing simultaneously different actions, the players are constantly moving, there are occlusions, the camera itself is moving. The proposed method is able to simultaneously recognize the actions of multiple players using pose estimation, tracking, and LSTM for action classification.

Acknowledgements

The greatest thanks for my advisor, who supported me a lot during work on a bachelor thesis. I am grateful to Ukrainian Catholic University to all the knowledge I got and for the scholarship for studying here. I want to thank my family for all the support and understanding. I am sending love to my dear friends, who motivate me to be a better version of myself.

Contents

List of Figures

List of Tables

List of Abbreviations

RNN	**R**ecurent-**N**eural **N**etwork
CNN	**C**onvolutional-**N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
SoTA	**S**tate **o**f **T**he **A**rt
MLP	**M**ulti**L**ayer **P**erceptron

Dedicated to my pillow, you saw my greatest happiness and my tears even though you are still there for me.

Chapter 1

Introduction

In today's world, we can use technological advancement to help people with their work or at least make their life easier. For example, it is hard to compare players on a basketball match, because we do not have a single value or some small set of metrics that can be taken quickly from the video. Also, it is crucial to know that the player is improving his technique or understand what needs to be improved. The solution to such a problem is rather complicated; that is why we choose to work on one part, action detection for each player. We have a significant amount of data available; there are a lot of basketball games in proper resolution on YouTube. It is crucial to have a good dataset, so the model will learn the right patterns and give the right prediction despite having a challenging task.

Action detection in a team sport is an exciting and challenging topic. Some challenges, we have to face:

- there are several objects of interest, which performing different activities as running, walking, dribbling (running and bouncing the ball), etc.
- the actions are pretty similar: walking, running, etc and person's poses do not differ much on these activities.
- the camera movements: the person can see that the camera is moving, while our approach does not have this information, this may be added in further work.
- there are two teams, and all team members have the same uniform (this is a challenge for player tracking)
- they often collide (for example before making the shot, while the shot class is crucial to distinguish) that it is nearly impossible to distinguish the whole body.
- there may be other people on background; their poses are not interesting for us.

What is easier in our task in contrast to the action detection in general:

- the environment is similar through different videos
- we have a smaller amount of classes to distinguish. We have six categories, while UCF101 [25] dataset has 101 classes, MultiTHUMOS [33] has 65 classes.

Comparing to the typical action recognition task that works on the level of sequences of entire video frames, our solution should work on the level of moving bounding boxes, without using RGB frames.

Chapter 2

Background

2.1 MLP

Multilayer Perceptron is a deep artificial neural network, that consists of only fully connected layers. The fully connected layer is a layer, in which all previous layer neurons' are connected with some weight w_i to all next layer's neurons and there are not any connections in the layer. There need to be three or more layers.

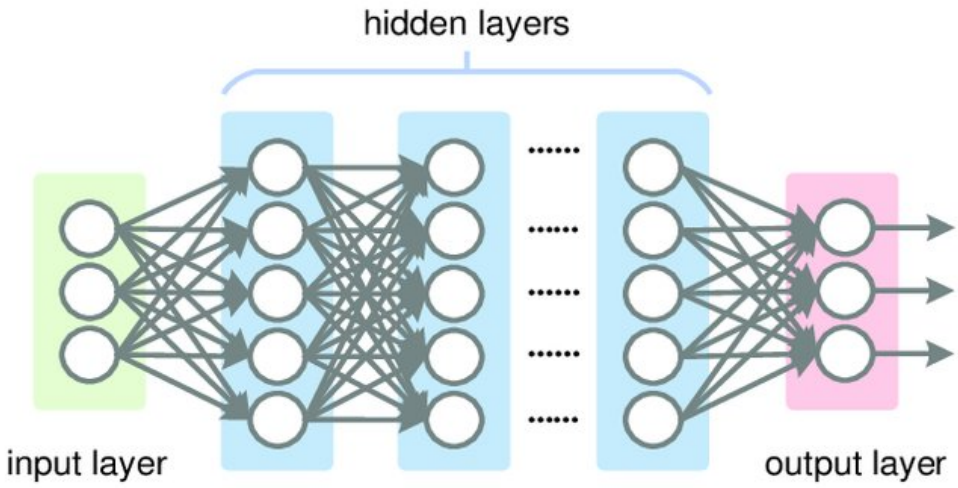

FIGURE 2.1: MLP architecture [30]

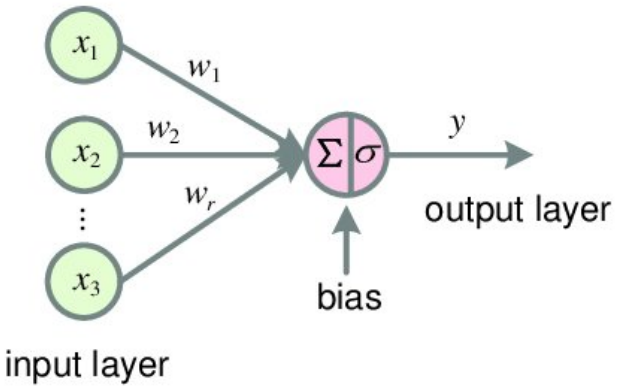

FIGURE 2.2: A mathematical model of neuron [30]

The one neuron takes as input x_i multiply it by weights w_i, add bias b, summarize all. The last step is sigmoid activation to give prediction y.

$$y = \sigma(\sum_i (w_i x_i) + b)$$

The main drawbacks are:

- with a growing number of neurons, the number of weights grows very fast;
- vanishing gradient;
- slow convergence;
- falling into a local minimum.

2.2 RNN

Recurrent Neural Network is used to process sequences of input data and output depends on both current input and state, which contains the memory of past inputs. In traditional Neural Networks, all the inputs are independent of each other. RNN is similar to human's memory, as we read we understand sentence based on words we read and recognise.

Another way of thinking about RNN is to understand it as a looping layer or a sequence of NNs. This architecture allows old information to be remembered through the process of learning new data.

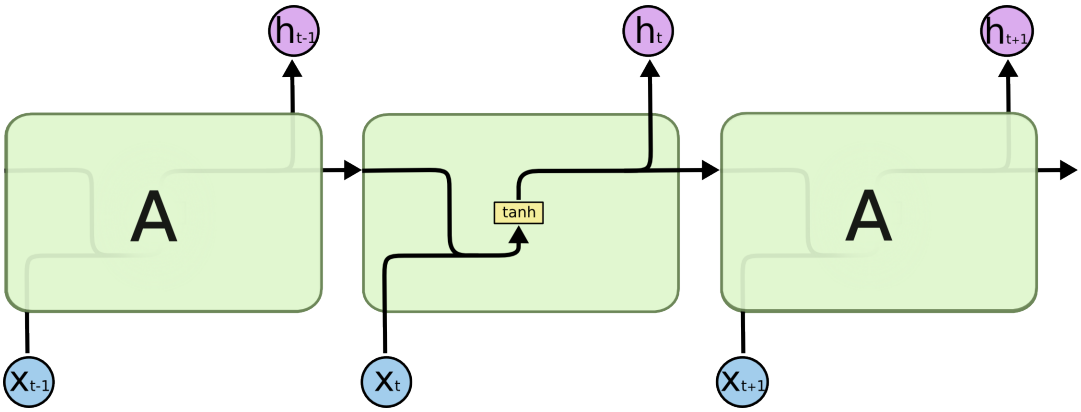

FIGURE 2.3: RNN architecture [29]

We can see that structure of RNN is rather simple. The output h_t is a combination of the information from previous cell h_{t-1} (cell is one piece of the chain) and new data x_t.

$$h_t = \tanh(W \cdot [h_{t-1}, x_t] + b)$$

2.3 LSTM

Long Short Term Memory (LSTM) is a Recurrent Neural Network (RNN) architecture that is capable of learning long-term dependencies [10]. LSTM is widely used in different areas of research, where we can find the sequence, as the action is some

sequence of frames or sequence of words in a sentence.

The RNN has vanishing gradient problem for long input sequences. RNN is unrolled into a feed-forward net with multiple layers (one layer for a one-time step). When the gradient is passed back multiple times for each time step, it tends to vanish (sometimes explode), the same way as it happens in the ordinary neural network with a large number of layers. LSTM solves this problem having three gates.

LSTM has a chain structure as all RNN models. The most interesting about this model is that it has various gates, which has a different purpose. The gates are NNs that decide which information is important and needed to be passed to the cell, and which is not – so, we can get rid of it. The LSTM has two types of memories: cell state C_t which acts as long term memory, and hidden state h_t which acts as working memory (a state at current time step). Unlike LSTM, RNN has just hidden state.

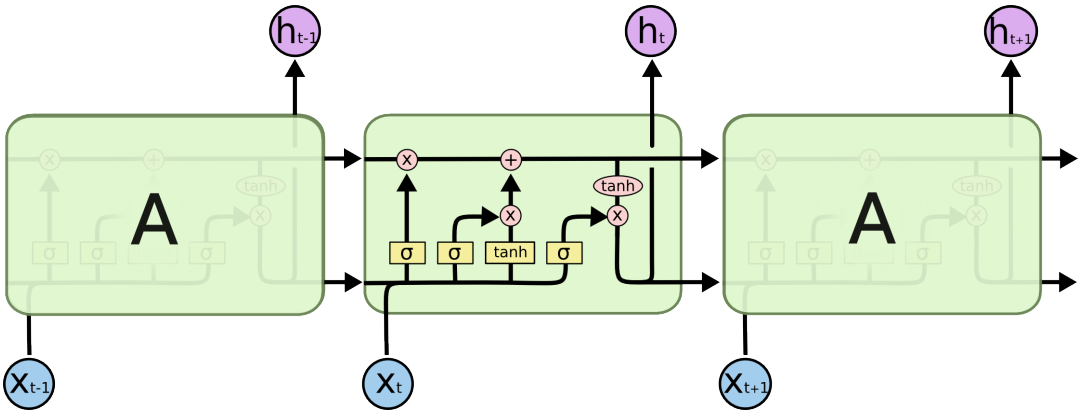

FIGURE 2.4: LSTM architecture [29]

The first gate (it is located in the bottom left) is Forget gate, here we choose which part of the information we need to ignore – forget from the previous hidden state h_{t-1} (a bottom line from the previous cell), based on input x_t. We use the sigmoid here to be able to get rid of the part of the data.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The next gate is Input gate that chooses, what we need to update in a cell state C_t, knowing new input information x_t. The output after the sigmoid is i_t. The sigmoid helps to carry only important values. While using tanh we rearrange values in -1 to 1 range to show the importance of each value.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The cell state is a sum of the previous cell state C_{t-1} multiplied by output from Forget gate f_t, then added output of Input gate $i_t * \widetilde{C}_t$, the values that are scaled based on importance. The cell state C_t is the upper line that goes into the next cell.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$$

The last gate is Output gate, where we choose the information that hidden state should hold h_t, as well as, the output from our cell h_t (the arrow out of the cell

pointing up). Here we choose only relevant values to output based on the previous hidden state h_{t-1} and the input x_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

2.4 BiLSTM

Bidirectional Long Short Term Memory [8] is a combination of two LSTMs, where one is taking information as it is (from start), while other – start from the sequence's end. This way model gets a better understanding of the data. It was proved that they outperform unidirectional LSTM in many research directions. The main disadvantage of such an approach, the whole sequence is needed to be passed to the model at the start.

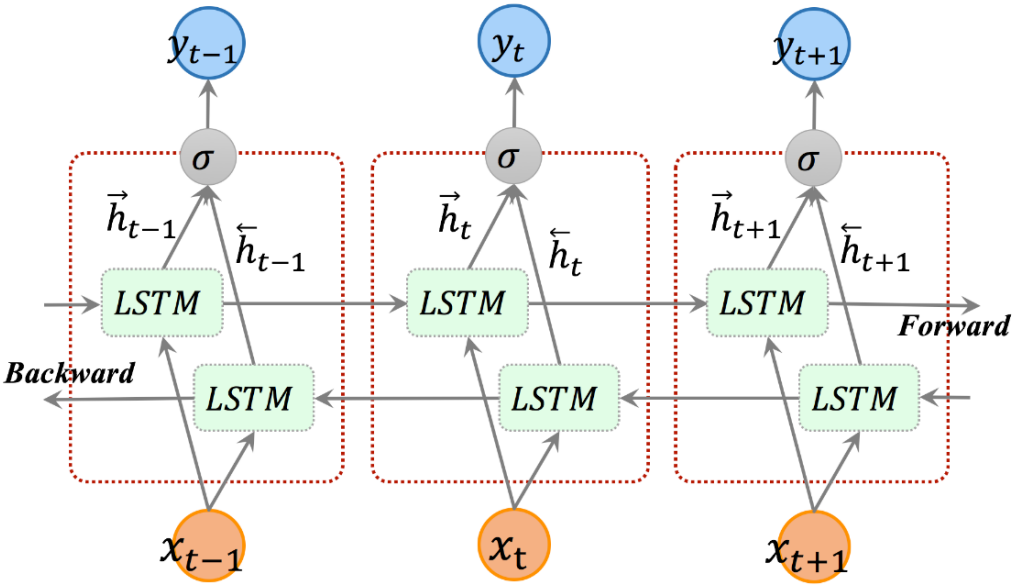

FIGURE 2.5: BiLSTM architecture [3]

The output y_t is the combination of hidden states of forward $\overrightarrow{h_t}$ and backward $\overleftarrow{h_t}$ models. The combination function may be concatenation, summation, multiplication or taking average.

$$y_t = \sigma(\overrightarrow{h_t}, \overleftarrow{h_t})$$

2.5 Human Pose Estimation

Human Pose Estimation gives the coordinates of the body joints of one or multiple people. It is widely used in action detection, animation tasks.

A more challenging task is the multi-person pose estimation, that can be solved with two approaches:

- top-down approach: try to detect people on video and then find the pose.
- bottom-up approach: find all body's parts and then connect them to find the pose.

It is hard to measure which approach is better as it mainly depends on detection function and grouping algorithm, that was chosen.

Chapter 3

Related work

Action detection task has a great variety of approaches. Some of them use different CNN models [11, 16, 28, 35], while others use RNN models [1, 24, 27]. Also, there exist solutions that use the advantage of combining both approaches [4, 18, 20, 32]. [24] approach model consists of the main LSTM network, the spatial attention subnetwork, and the temporal attention subnetwork. The network automatically chooses crucial joints between the frames using a spatial attention model, then it gives importance score to the frames using the temporal attention subnetwork. An action prediction goes from combining outputs of main LSTM network and the temporal attention subnetwork. The accuracy on NTU dataset [23] with CrossSubject is 73.4% and with Cross-View is 81.2%.

3.1 Pose-Based Action Recognition

Some approaches rely on a person's pose directly [1, 11, 16], while other builds CNN for detecting the pose [4, 6]. [16] is one of the small amount approaches that meet real-time restriction, as they use Yolo v3 [21] for people detection then pose is estimated using optical flow [31], for action classification is used simple CNN.

The [1] approach meets real-time performance. They propose two pipelines ActionXPose-basic and ActionXPose-advanced. The first step of both approaches is dealing with missed data. If the pose doesn't have enough landmarks, it will be discarded. Some landmarks can be estimated by knowing the location of other landmarks. Nest step is pose centring and scaling, which gives more robust pose representation. In the advanced approach are additional steps as spatial information analysis, which learns generalised poses from training data, temporal information analysis, which looks for changes in the pose through the time, Spatio-temporal embedding allows to measure the distance between generalised poses and the one in the input for each frame. Final sequence classification is done by MLSTMFCN [14]. They achieve 99.04% accuracy on KTH dataset [22].

[4] get main information about human's pose from CNN, after that they group semantically-related joints, this way they try to get richer information about the pose. For each body part, they use LSTM hidden state and CNN predictions to get the pose's features. Then, they fuse the information from all the body parts and give it to LSTM, which predicts action. The big benefit of this approach is that test data can be without annotated joints, as model learned robust features for all body parts.

3.2 3D Convolutional Neural Networks

3D Convolutional Neural Networks [26] are designed to work on video-level, not on frame-level. As input it gets volume – the video, the output is a volume, as well. A similar approach in this direction was done on the base of Resnet architecture [9]. The main drawbacks of these methods are a large number of parameters, they tend to overfit; expensive an interference time. The accuracy is lower than in other approaches.

3.3 Multi-stream Fusion

The multi-stream CNN [7] is used for such task [11, 18, 28, 35], here they extract visual features using RGB and motion features using optical flow. In [11] approach in addition to stated above features, they use preprocessed pose prediction, which goes into the proposed 3D temporal pose CNN. The next step is a multi-dimensional fusion, which allows taking advantage of both 2D and 3D CNN feature maps. They get awesome results on PennAction dataset [34] the accuracy is 97.6%.

[28] multi-stream CNN consist of three two-stream networks, the first network input is bounding box of human; the second region is selected from the first one using their motion saliency measure; the third input is entire RGB image. From these three regions, three motions streams are formulated. After extracting descriptors from streams they are fused for the action recognition task.

3.4 Action detection in team sports

Action recognition in videos with many people moving is a challenging task. As different players perform different actions and in some sports, the players and the ball are very fast, so they may be blurry. It is common that the camera is moving through the video; it sometimes may zoom in or out. There exist action detection researches for different sports like hockey [6], football [27], basketball [20], etc. Some approaches have a goal to distinguish both sports actions and everyday actions [18, 32].

[6] approach tracks player and crop a frame with the player in centre, then it used the stacked hourglass network [17] to get a set of heatmaps of body parts that can be combined to general pose. Next component is feature transformer that gets information from heatmaps and generates a better pose representation. It helps to distinguish actions based on the pose. The action is predicted by six fully connected layers, the number of layers, as well as, number of neurons are chosen empirically. Hockey is a fast game and it has a lot of moments when people are occluded by other people. Another difficulty is that people are wearing protective equipment, which makes pose estimation task even more challenging. This approach has an accuracy of 65.47% for four classes.

The [18] approach pays attention to the sequence of events that always happen in some order. They build a pipeline that allows learning temporal action relations through the video. From each frame, they extract information – action probability, using CNN. The next step is the temporal structure filter, which is proposed in this

paper, that gives the information where in the video is relevant segments for frame-level event detection. They learn a set of temporal structure filters, that is smaller than the number of classes, as they assume that some classes will share same filters or use a combination of given filters. Then, using information from several frames (the super-event representation) and data from CNN, they predict the class for each frame. This approach was SoTA; on MultiTHUMOS [33] mAP is 36.4%.

The [19] approach present new Temporal Gaussian Mixture (TGM) layer. They use it on top of sequence representation. This layer allows learning features while having a smaller number of teachable parameters. Their model doesn't have a separate set of Gaussian distributions per class, instead, it has a combination of Gaussians. The approach is SoTA, while using super-events and TGM layers and has 46.4% accuracy.

In [20], they try to find the key person and classify action, while in this work we will mainly focus on predicting action for each player. To represent each frame they use the activation of the last fully connected layer of the Inception7 network; features that describe a person's appearance and spatial information about the player. The event classification is done by $BLSTM$ in combination with $LSTM$. Each player's track is an input to separate $BLSTM_{track}$, hidden state of which is used in main $LSTM$, which represent the state of the event. Also, there is another $BLSTM_{frame}$ that extract information from the whole frame and output the data to the main $LSTM$. The attention model is using $BLSTM_{track}$'s hidden state to find an important player.

Chapter 4

Recognition Pipeline

There are a vast majority of approaches, a big amount of them need huge datasets and a lot of computation resources. Our approach is simple, at the same time, it is efficient.

FIGURE 4.1: The model's prediction for one frame

The result of our work is shown on Figure 4.1. In other words, the result of our approach is the model, that is able to predict action for each player.

4.1 Pose prediction

For Action Recognition we need to have the sequences of player's poses through some amount of time. We selected 17 as the length of the sequence, which corresponds to the time duration of 0.68 sec (25 FPS) or 0.57 sec (30 FPS). We have found that this duration should be sufficient to recognise player actions in basketball, that is a fast and dynamic game, even though a player may change its action during the 1-second interval (e.g. RUN -> RECEIVE_BALL -> WALK). The prediction of LSTM corresponds to the centre element of the sequence, that is why we have 8 frames before the centre element, and 8 frames after. We used Alpha pose [5] for pose prediction and then SORT tracker [2] to distinguish players one from another. The Alpha pose works well on videos with many people, but it doesn't track people through the frames. SORT is a fast and elegant way to track players through the video.

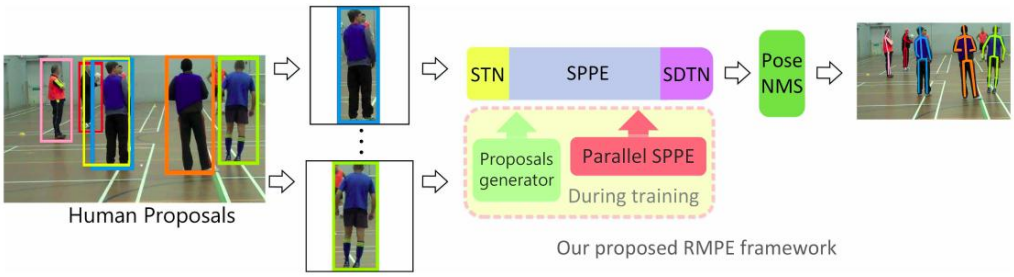

FIGURE 4.2: Alpha pose pipeline [5]

4.1.1 Alpha pose

We use Alpha pose [5] for pose estimation. The alpha pose is a framework, that predicts human's pose and outputs the key points of main joints (17 key points). It is a top-down method (find the person and predict the pose), the accuracy of which highly depend on people detector.

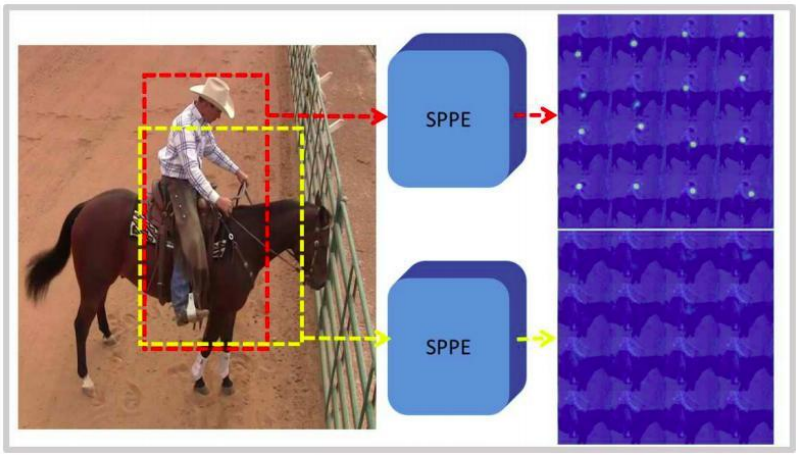

FIGURE 4.3: The problem that Alpha pose is solving [5]
The problem of low confidence in bounding boxes (red - ground truth, yellow - prediction).

They focus on improving the search of bounding boxes, using Symmetric Spatial Transformer Network (SSTN) attached to Single-Person Pose Estimator (SPPE) [17]. Spatial Transformer Network (STN) [12] helps to find high-quality features from bounding boxes predicted by SPPE.

Some predicted poses may not be correct, to automatically distinguish them the researches use pose non-maximum suppression (NMS). Firstly, most confident poses are selected, others that are similar to most confidence are also accepted.

4.1.2 SORT

SORT [2] is real-time multiple object tracker, as input goes bounding boxes for different frames, as output we get the id for each bounding box. This tracker follows tracking by detection paradigm. It uses only bounding boxes, no appearance information is used.

The model they constructed uses Kalman Filter [13] and Hungarian method [15], as both are extremely efficient. They construct a linear constant velocity model, that approximate the object's movement through the frames, this model is not affected by other objects and camera movement. The object state is described by this formula:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$$

Where u and v are the horizontal and vertical location of the object's centre. Variables s and r are area (scale) of the object and aspect ratio of the object. When we find a detection that corresponds to the target, we update the state, using a Kalman filter [13] for finding optimal velocity parameters. The next step is finding bounding boxes with maximum IoU over the neighbour frames, using Hungarian method [15]. Then ids for each bounding box are assigned.

4.2 Action Recognition

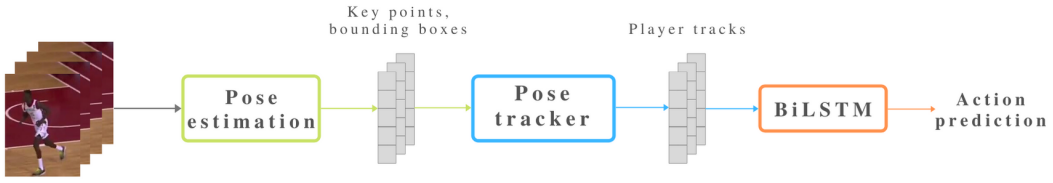

FIGURE 4.4: BiLSTM pipeline

As was stated before, we use pose estimation and multiple object tracker, from which we get sequences of key point vectors for some time. This player track will be an input into BiLSTM or another model, which will predict action for one person.

Our model should learn distinguishable features for each class, even if our classes are similar and we do not have many examples. About these challenges, we will talk in the next sections.

Chapter 5

Dataset

We create our own dataset of basketball match videos with good resolution, choose the part of the game, where we can distinguish a player's action. We use 6 classes: RECEIVE_BALL, THROW (the ball), DRIBBLE (bouncing the ball), RUN, WALK, NO_ACTION (another action from all stated before). We use 37 videos for data annotation, 11 from them are taken from Youtube, rest from another source. We prepare video before annotating. We run Alpha pose for each frame. The next step is to give all bounding boxes from different frames to SORT tracker and get the player's track. It is important to select an action in the middle, as for example for RECEIVE_BALL the model will learn arms position before and after this fast action.

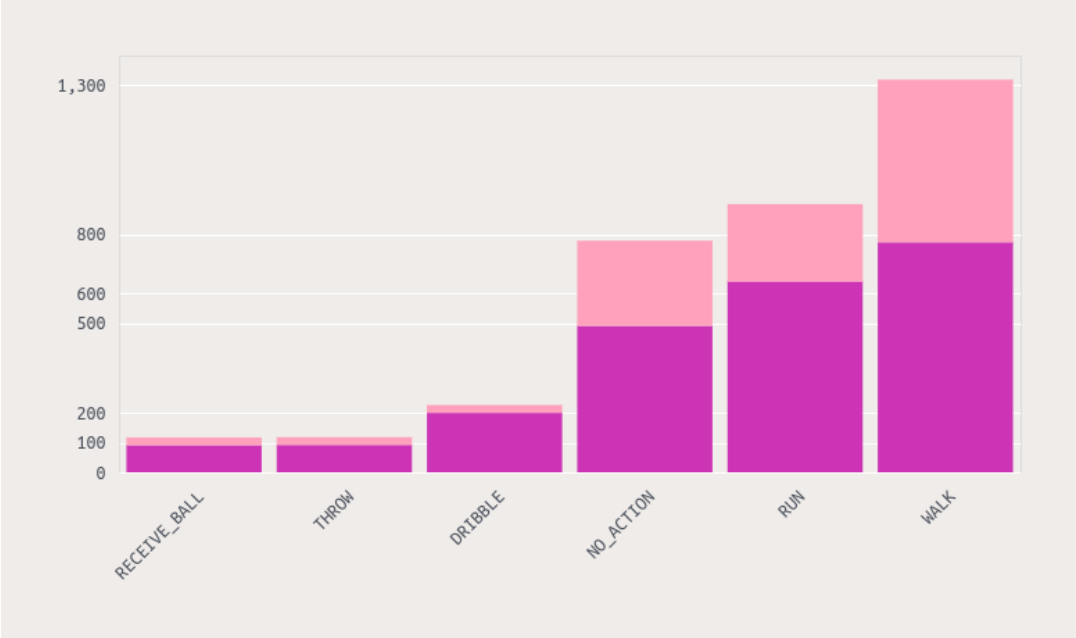

FIGURE 5.1: The classes distribution among the dataset.
The pink colour is data that was added, purple – starting data.

As we can see on the Figure 5.1 we have unbalanced data, where all ball related actions have a small number of examples. One more challenge is that actions run and walk are pretty similar, in addition, some examples are in between run and walk. Another challenge is that dribble is similar to walk and run, the only difference is moving hand, as we don't use RGB image. The recognition of actions that involve ball (dribble, throw, receive the ball) may be significantly improved by using additional ball detector or tracker, at present, we do not use ball location.

5.1 Data Preprocessing

The keypoints returned by Alpha pose are in pixel coordinates, thus their values depend on frame position and scale (size of the bounding box). We standardize keypoints relatively to bounding box centre and size with the goal to make keypoints position and scale-invariant. So we will have x values between -1 and 1 and y values between -1 and 1, we think this will give a model better explanation of where key points are located in relation one to another. As we are using Alpha pose we have a score for each keypoint it predicted. If this score is less than 0.1, we will put this point into the centre with coordinates (0, 0), so bad keypoint prediction will less affect a player's pose and general performance. In future, feature engineering may be done on this part of the solution.

The labelled videos had different fps 25, 30 and 50. While 25 fps and 30 fps are pretty similar, the videos with 50 fps are different. That's why we downsample videos with big fps to 25 fps, and this is named additional data and is pink on the Figure 5.1 and on the Figure 5.2. The additional data was split to train, validation and test.

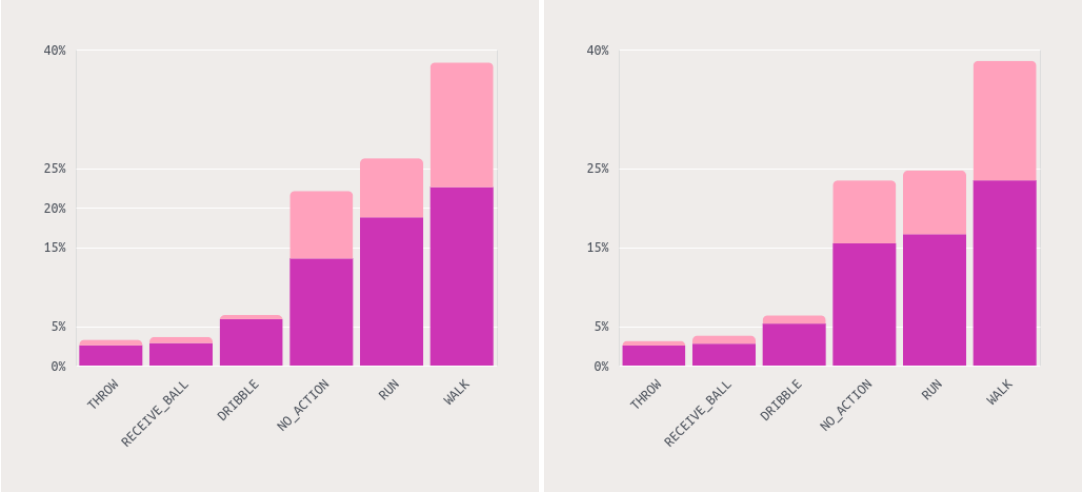

FIGURE 5.2: The percentage of examples in train (left) and test (right). The pink colour is data that was added, purple – starting data.

The dataset was split into three categories: train, validation and test. For the train, we use 2073 sequences, for the validation – 520 and for the test – 866.

Using additional pink data on Figure 5.2, the accuracy increased from 64% to 75%. This shows that our model benefits from a bigger number of training examples.

We augment 40% of training data using vertical transform to have more data. As a result, we have 2902 training examples. In future work, we can augment data adding some noise to the coordinates of key points.

Chapter 6

Experiments

The LSTM is an awesome architecture to work with classification problem on time series data. When a player is performing a certain action, his body parts (legs, arms) follow motion patterns. Thus it is natural to choose the LSTM model to perform such a task. We would like to try an alternative approach to compare, and decided to choose MLP architecture. As a result, we have several hypotheses to test:

- BiLSTM will perform better than the MLP approach because it uses the information that it gets the sequence as input.
- BiLSTM will perform better than LSTM, as it read sequence from start to end and from end to start.
- Classes RUN vs WALK, WALK vs NO_ACTION will be difficult to distinguish.

6.1 Implementation details

The neural networks were implemented on Keras with TensorFlow backend. During the project, we were using Python, Jupiter, Google Colab, TensorBoard, Pandas etc.

As loss function, we use categorical cross-entropy. We use Adam optimizer. As accuracy, Keras uses categorical accuracy, that calculates the frequency of the predictions matches one-hot labels. The activation functions are ReLU for all Dense layers, except the last, there we use Softmax.

6.2 Results

Our first guess was to try BiLSTM model with 100 units per LSTM and then add some Dense layers, but it did not work, the validation accuracy was fluctuating all the time, during the training. The model starts overfitting and validation accuracy was near 73%. There are many options for reducing overfitting. One option of them is to reduce the number of parameters, thus we reduced the number of LSTM units from 100 to 32 (in case of bidirectional LSTM this is multiplied by 2, which gives 64 units as you can see in the diagram 6.1). Another option is adding the Dropout layers.

The BiLSTM model description is on Figure 6.1, loss and accuracy per epoch is on Figure 6.2. We can see that validation loss stop falling on 50th epoch and start fluctuating. It signalises us that the model starts overfitting on train data.

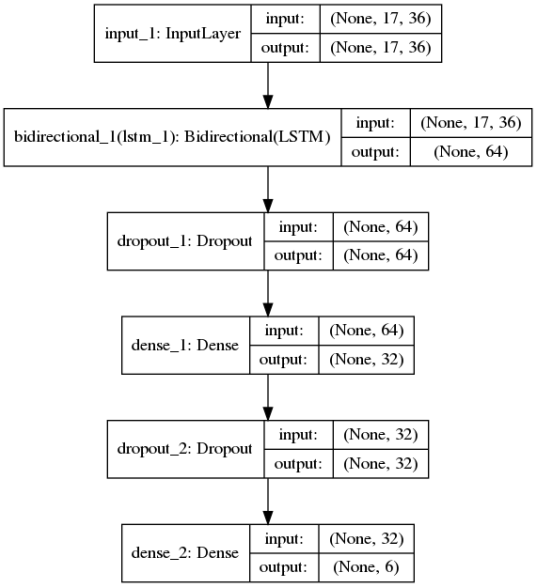

FIGURE 6.1: BiLSTM model description

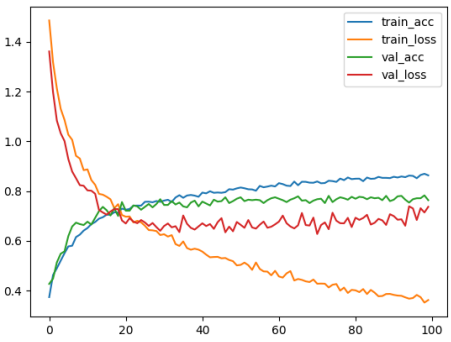

FIGURE 6.2: BiLSTM training procedure

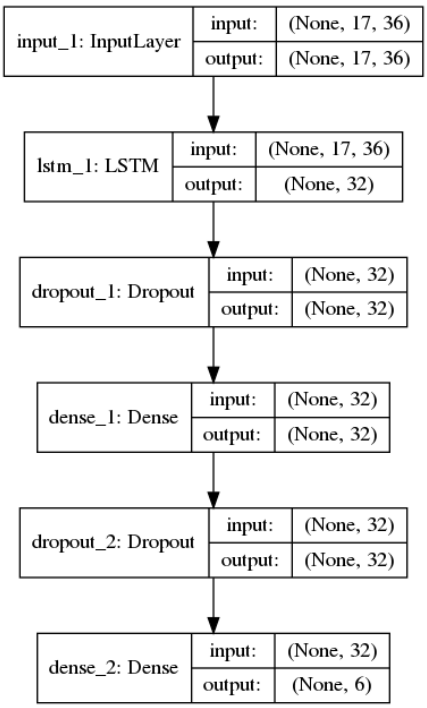

FIGURE 6.3: LSTM model description

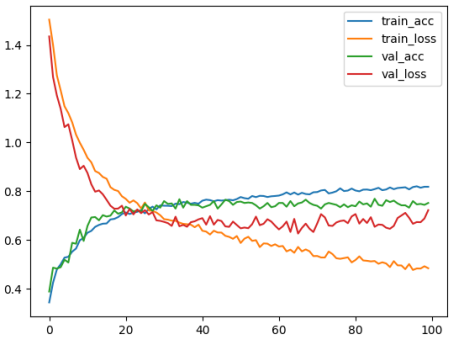

FIGURE 6.4: LSTM training procedure

For the LSTM model happens something similar to described above. The model architecture is described on Figure 6.3 and loss on Figure 6.4. The model starts overfitting and the loss for validation data stop decreasing on 60th epoch.

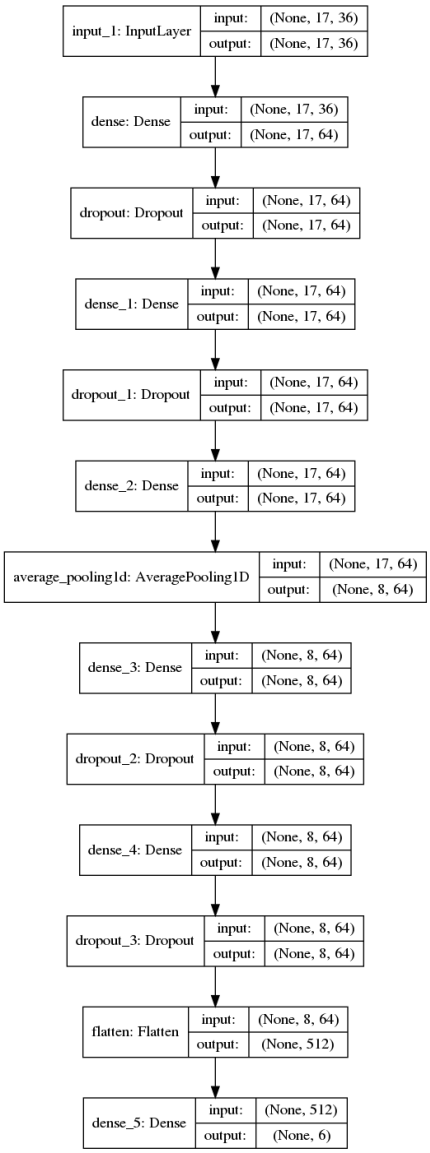

FIGURE 6.5: MLP model description

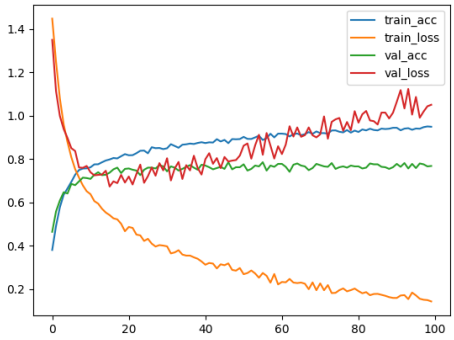

FIGURE 6.6: MLP training procedure

We tried many different configurations of the MLP model, changed a number of layers and a number of filters to get the best accuracy. A vast majority of the models we tried was having low accuracy combined with high overfitting. The model demonstrated on Figure 6.5 has the best performance. The training process is shown on Figure 6.6. While the LSTM and BiLSTM validation loss saturated at a certain level, here validation loss started to grow after some epoch. As it can learn more deep features, it overfits even more than others.

The dataset we use for training is rather small. We can have better results if we use bigger and more balanced dataset.

BiLSTM and LSTM models take advantage of knowing how the pose develops over time; the MLP model has information as a set of values. The BiLSTM model outperforms both LSTM and MLP models. However, the number of trainable parameters for LSTM model is two times smaller and the accuracy is only 2% less than the BiLSTM model, but it still is better than the MLP model's accuracy. The LSTM and BiLSTM models have better training curve, so they learned more generalized features.

Architecture	# of parameters	Accuracy
LSTM	10,086	78.52
BiLSTM	19,942	**80.60**
MLP	22,086	77.25

TABLE 6.1: Results on test data for different architectures

It is interesting to look at the confusion matrix of our models to see, which classes are less distinguishable.

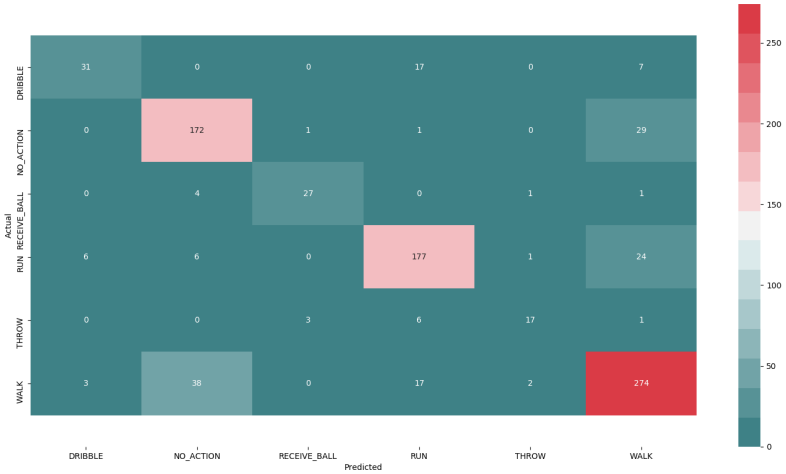

FIGURE 6.7: Confusion matrix for BiLSTM model

The classes WALK and NO_ACTION are easy to misclassify as there is no much difference in slow WALK and NO_ACTION at all. The camera is moving and the model may think that the player is moving, while in fact it can stand still or move only his arms. This can be improved by calculating global camera motion in x and y direction between frames (e.g. using optical flow) and adding this as an additional input to our model.

The other challenge is with fast WALK and RUN, as they look similar.

One more problem is distinguishing RUN and DRIBBLE. We can give better predictions when we know the location of the ball in reference to the chosen player.

All in all, the diagonal has big values, showing that the model is well trained.

Chapter 7

Conclusion

To sum up, we did research on methods of action recognition for multiple players, that differs from typical approaches for video level action recognition. We annotated new small dataset with six actions. We propose the solution with 80.6% accuracy, that seems to be the maximum for our small dataset. We can have accuracy improvement if we increase dataset size. During work on this project, we tried a different model's architectures and configurations. We get a better understanding of the training process; we tried various techniques for dealing with overfitting.

In further work, we will try our approach on heatmap images of a person's joints. We can try feature engineering on our data, it may help with the key points, we got from Alpha pose, that have a low score. We may add some parameters: relative motion between joints, global camera motion and ball position. It will be interesting to compare the proposed solution to Multi-stream Fusion models trained on our data.

Bibliography

[1] Federico Angelini et al. *ActionXPose: A Novel 2D Multi-view Pose-based Algorithm for Real-time Human Action Recognition*. 2018. arXiv: 1810.12126 [eess.IV].

[2] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)* (2016). DOI: 10.1109/icip.2016.7533003. URL: http://dx.doi.org/10.1109/ICIP.2016.7533003.

[3] Zhiyong Cui et al. *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*. 2018. arXiv: 1801.02143 [cs.LG].

[4] W. Du, Y. Wang, and Y. Qiao. "RPAN: An End-to-End Recurrent Pose-Attention Network for Action Recognition in Videos". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3745–3754.

[5] Hao-Shu Fang et al. *RMPE: Regional Multi-person Pose Estimation*. 2016. arXiv: 1612.00137 [cs.CV].

[6] M. Fani et al. "Hockey Action Recognition via Integrated Stacked Hourglass Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 85–93.

[7] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. *Convolutional Two-Stream Network Fusion for Video Action Recognition*. 2016. arXiv: 1604.06573 [cs.CV].

[8] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition." In: Jan. 2005, pp. 799–804.

[9] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. *Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition*. 2017. arXiv: 1708.07632 [cs.CV].

[10] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[11] Yi Huang, Shang-Hong Lai, and Shao-Heng Tai. "Human Action Recognition Based on Temporal Pose CNN and Multi-dimensional Fusion: Subvolume B". In: Jan. 2019, pp. 426–440. ISBN: 978-3-662-53906-4. DOI: 10.1007/978-3-030-11012-3_33.

[12] Max Jaderberg et al. *Spatial Transformer Networks*. 2015. arXiv: 1506.02025 [cs.CV].

[13] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. URL: https://doi.org/10.1115/1.3662552.

[14] Fazle Karim et al. "Multivariate LSTM-FCNs for time series classification". In: *Neural Networks* 116 (2019), 237–245. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.04.014. URL: http://dx.doi.org/10.1016/j.neunet.2019.04.014.

[15] H. W. Kuhn and Bryn Yaw. "The Hungarian method for the assignment problem". In: *Naval Res. Logist. Quart* (1955), pp. 83–97.

[16] Dennis Ludl, Thomas Gulde, and Cristóbal Curio. *Simple yet efficient real-time pose-based action recognition*. 2019. arXiv: 1904.09140 [cs.CV].

[17] Alejandro Newell, Kaiyu Yang, and Jia Deng. *Stacked Hourglass Networks for Human Pose Estimation*. 2016. arXiv: 1603.06937 [cs.CV].

[18] AJ Piergiovanni and Michael S. Ryoo. *Learning Latent Super-Events to Detect Multiple Activities in Videos*. 2017. arXiv: 1712.01938 [cs.CV].

[19] AJ Piergiovanni and Michael S. Ryoo. *Temporal Gaussian Mixture Layer for Videos*. 2018. arXiv: 1803.06316 [cs.CV].

[20] Vignesh Ramanathan et al. *Detecting events and key actors in multi-person videos*. 2015. arXiv: 1511.02917 [cs.CV].

[21] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[22] C. Schuldt, I. Laptev, and B. Caputo. "Recognizing human actions: a local SVM approach". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. 2004, 32–36 Vol.3.

[23] A. Shahroudy et al. "NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1010–1019.

[24] Sijie Song et al. *An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data*. 2016. arXiv: 1611.06067 [cs.CV].

[25] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV].

[26] Du Tran et al. *Learning Spatiotemporal Features with 3D Convolutional Networks*. 2014. arXiv: 1412.0767 [cs.CV].

[27] T. Tsunoda et al. "Football Action Recognition Using Hierarchical LSTM". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 155–163.

[28] Zhigang Tu et al. "Multi-stream CNN: Learning representations based on human-related regions for action recognition". In: *Pattern Recognition* 79 (July 2018), pp. 32–43. DOI: 10.1016/j.patcog.2018.01.020.

[29] *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[30] Tianqi Wang et al. "Deep Learning for Wireless Physical Layer: Opportunities and Challenges". In: *China Communications* 14 (Oct. 2017), pp. 92–111. DOI: 10.1109/CC.2017.8233654.

[31] Bin Xiao, Haiping Wu, and Yichen Wei. *Simple Baselines for Human Pose Estimation and Tracking*. 2018. arXiv: 1804.06208 [cs.CV].

[32] Serena Yeung et al. *End-to-end Learning of Action Detection from Frame Glimpses in Videos*. 2015. arXiv: 1511.06984 [cs.CV].

[33] Serena Yeung et al. *Every Moment Counts: Dense Detailed Labeling of Actions in Complex Videos*. 2015. arXiv: 1507.05738 [cs.CV].

[34] W. Zhang, M. Zhu, and K. G. Derpanis. "From Actemes to Action: A Strongly-Supervised Representation for Detailed Action Understanding". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 2248–2255.

[35] Mohammadreza Zolfaghari et al. *Chained Multi-stream Networks Exploiting Pose, Motion, and Appearance for Action Classification and Detection*. 2017. arXiv: 1704.00616 [cs.CV].