Spatio-Temporal Based Table Tennis Hit Assessment Using LSTM Algorithm

Kadir Aktas[1], Mehmet Demirel[2], Marilin Moor[1],
Johanna Olesk[1], Gholamreza Anbarjafari[1,3]
[1]University of Tartu, Estonia
[2]University of Manchester, United Kingdom
[3]PwC Advisory Finland, Itämerentori 2, 00180 Helsinki, Finland
(kadir.aktas,marilinm,johanna.olesk,shb)@ut.ee
mehmet.demirel@student.manchester.ac.uk

ABSTRACT

In these working notes, we present our approach and results for Mediaeval 2020 Sports Video Classification Task [6]. We implemented a multi-stage pipeline with LSTM-based network. In the developed approach, firstly, the frames are extracted, sampled and resized. Then, considering that the stroke type has three different parts, each part is labelled and predicted separately. In order to obtain the predicted stroke type, the predictions for each part are fused together.

1 INTRODUCTION

Sports action recognition is a well-studied research topic due to the wide application area and commercial value. Although many methods are developed for different sports tasks [2, 9], the challenge of performing more precise analysis still remains open, especially for low variance classification tasks such as table tennis stroke type classification. To address this claiming, Martin et al. collected TTSTROKE-21 dataset [8] and the Mediaeval 2019 and 2020 Sports Video Classification Task were created [6, 7].

In this paper, we present a multi-stage spatio-temporal recognition method using long short-term memory (LSTM) [4, 13, 15] based network. Our architecture predicts the final label in three stages. In the first stage, the position (serve, offensive, defensive) is classified. In the second stage, the hand orientation (forehand, backhand) is classified. Finally, in the third stage, the hit technique (flip, hit, push, block, loop, topspin, backspin, sidespin) is predicted using one of the 3 different models. The first model classifies serve techniques. Second and third models classify offensive and defensive techniques, respectively. Lastly, in order to obtain the final stroke type, a fusion of labels

2 RELATED WORK

Recently there has been an increase in the number of studies in table tennis stroke type recognition from videos. Martin et al. have collected TTSTROKE-21 dataset and proposed a Twin Spatio-Temporal Convolutional Neural Network (TSTCNN). Their network uses an RGB image sequence and optical flow calculations as an input. They extract and resize the frames to (320×180) for each stroke in order to use them as input data [8]. Instead, we resize the frames to (120×80) to increase the processing speed.

Sriraman et al. present another approach which extracts features using Convolutional Neural Network (CNN) and applies them to a spatio-temporal model [10]. They use VGG16 network [11] as the feature extractor and apply Long Short Term Memory (LSTM) [4] layer on the extracted features. They use 25 frames, which are sampled by a varying rate, per each move. In our work, we use 21 frames based on centroids of the k-nearest neighbour method. Also, we extract spatio-temporal features only and do not use a CNN-based feature extractor.

3 APPROACH

To face the challenge of a high number of classes and low variance between them, we designed a multi-stage approach. We divided the initial 20 labels into 5 groups (see Figure 1). In stage 1 and 2, the first and second parts of the final label are predicted. In stage 3, the third part of the final label is predicted, however, the prediction is done based on stage 1 results. For example, if the stage 1 predicts 'Serve' then in stage 3 the model which is trained for predicting one of 'Topspin', 'Sidespin', 'Backspin', 'Loop' is used. We used the same input and model structure for each stage, meaning, that we trained the same model for each label subset, for 5 times in total.

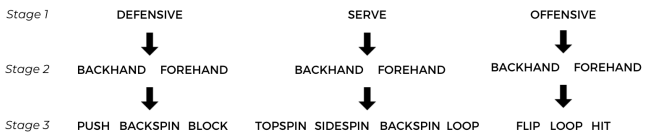

Figure 1: Labels splits

3.1 Data pre-processing

The dataset contains videos with 120 fps and resolution of (1920×1080). Considering that a single stroke has minimum of 100 frames [8], processing the data without resizing causes memory and time-related issues. So, to speed up the process and decrease memory restrictions we resize each frame to (120×80).

Each move in the dataset has a varying frame range. This means that we need to sample them in a fixed size as our model require a fixed input size. We sample 21 frames per move. This number is picked heuristically, however, we have tested that if the sample size

MediaEval'20, December 14-15 2020, Online

is too low, i.e. 7 frames per move, then the accuracy is decreased significantly. This way we boost the processing performance and provide a fixed input size to our model. Such approaches are well-known in the video indexing [1].

To decide on which frames to sample we use centroids of k-nearest neighbor (KNN) method. This method reflects the data distribution as the centroids are calculated using nearest neighbours [16]. We use RGB values of the resized images to compute KNN. And, We calculate 21 centroids and then sample 21 frames closest to each of the calculated centroids, respectively. Additionally, we flatten each frame in order to ensure they fit into our model.

3.2 Model

Our model uses RGB images as the input data without any prior feature extraction. In our model, firstly, the batch normalization layer is used to regularize the input data. This step processes the input batch by batch, subtracts mean and divides by standard deviation [5]. Then, two LSTM [4] layers are included in order to capture spatio-temporal features. These layers are constructed with unit numbers 128 and 32 respectively. Afterwards, a fully connected layer with 64 units is included to model the relation between features and the output. Each of these 3 layers is followed by a dropout layer at the rate of 0.2 in order to prevent overfitting [12]. Finally, an output layer with softmax activation is added to do the classification.

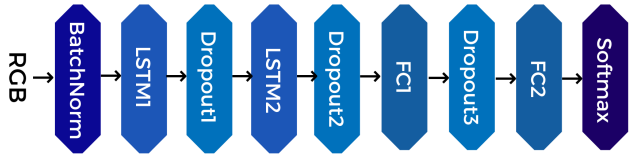

Figure 2: Model architecture

3.3 Training

Fully connected layers are initialized by using Glorot initialization [3]. We use categorical cross-entropy as the loss function and RMSprop as optimizer [14] with a learning rate of 0.0001. The training is done with batch size 8 in 30 epochs. 10-fold cross-validation is applied to prevent a biased data split.

We use the same model architecture and hyperparameters to train 5 different models. Each model has its own purpose, so they get trained with different subsets of the training data for different sets of labels (see Table **??**).

We split our data into train, validation and test splits by 0.6, 0.2, 0.2 proportions. Train and validation splits are used during model training. Test split is only used to test the trained model.

4 RESULTS AND ANALYSIS

Training results are shown in Table 1. We obtained 94.7% test accuracy for stage 1, i.e. classifier for 'Serve', 'Defensive', 'Offensive' labels. An accuracy of 98% was achieved for stage 2, i.e. 'Forehand', 'Backhand' classification. However, for stage 3 we got 80.1% accuracy, which is much lower than others. When combining the predicted labels in the final label, we obtained 78.1% stroke type prediction accuracy.

These results can be explained by a couple of factors. Firstly, the volume and distribution of the data affect the results. In stage 3 each label has considerably fewer data compared to the data numbers of labels in other stages. Also, especially in stage 3 labels, the data distribution is highly biased towards some classes, causing biased learning. Additionally, due to the nature of the task, stage 3 labels have less variance between each other compared to other stages. Lastly, since stage 3 is conditioned on the outcome of stage 1, some of the errors are caused by this outcome.

Table 1: Training Results

Stage	Validation Accuracy	Test Accuracy
1	91.4%	94.7%
2	98.7%	98%
3	82.8%	80.1%
Final	79.8%	78.1%

Our method got 9.32% accuracy on the run processed by MediaEval on a different test set. It was able to correctly predict classes of 33 samples out of 354. Our run results show that the method was able to achieve 50.85% for stage 1 prediction and 66.67% for stage 2 prediction. Although the accuracy for stage 3 is not published, it is obvious that the model had the lowest accuracy on stage 3 with a big gap (See Table 2).

Table 2: Run Result

Stage	Accuracy
1	50.85%
2	66.67%
3	N/A
Final	9.32%

Results of the MediaEval test run show that the model did not learn properly. Also, when we precisely analyze the results we see that the method is biased towards some classes.

5 DISCUSSION AND OUTLOOK

We obtained promising results during the training and validation, which assures there is no occurrence of overfitting. However, as the test results show, the model has failed to properly learn, i.e. was not able to generalise the learning. It is expected this can be addressed by having more labelled data in the training.

We also argue that the low variance between the classes and nature of the task causes the aforementioned challenge. Considering that a single class can be sampled in many ways for different players, i.e. right/left-handed or high/low experienced, we discuss that the dataset can be improved to increase the coverage of the classes as well as reducing the bias among the classes.

REFERENCES

[1] Maylis Delest, Anthony Don, and Jenny Benois-Pineau. 2006. DAG-based visual interfaces for navigation in indexed video content. *Multimedia Tools and Applications* 31 (10 2006), 51–72. https://doi.org/10.1007/s11042-006-0032-4

[2] Mehrnaz Fani, Kanav Vats, Christopher Dulhanty, David A. Clausi, and John S. Zelek. 2019. Pose-Projected Action Recognition Hourglass Network (PARHN) in Soccer. In *16th Conference on Computer and Robot Vision, CRV 2019, Kingston, ON, Canada, May 29-31, 2019*. 201–208. https://doi.org/10.1109/CRV.2019.00035

[3] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[5] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[6] Pierre-Etienne Martin, Jenny Benois-Pineau, Boris Mansencal, Renaud Péteri, Laurent Mascarilla, Jordan Calandre, and Julien Morlier. 2020. Sports Video Classification: Classification of Strokes in Table Tennis for MediaEval 2020. In *Proc. of the MediaEval 2020 Workshop, Online, 14-15 December 2020*.

[7] Pierre-Etienne Martin, Jenny Benois-Pineau, Boris Mansencal, Renaud Péteri, Laurent Mascarilla, Jordan Calandre, and Julien Morlier. 2019. Sports Video Annotation: Detection of Strokes in Table Tennis task for MediaEval 2019. In *MediaEval 2019 Workshop*.

[8] Pierre-Etienne Martin, Jenny Benois-Pineau, Renaud Péteri, and Julien Morlier. 2020. Fine grained sport action recognition with Twin spatio-temporal convolutional neural networks: Application to table tennis. *Multimedia Tools and Applications* (2020), 1–19.

[9] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. 2020. FineGym: A Hierarchical Video Dataset for Fine-Grained Action Understanding. (2020), 2613–2622. https://doi.org/10.1109/CVPR42600.2020.00269

[10] Vishnu K. Krishnan Bhuvana J Siddharth Sriraman, Srinath Srinivasan and T. T. Mirnalinee. 2019. MediaEval 2019: LRCNs for Stroke Detection in Table Tennis. (2019).

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

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[13] Martin Tammvee and Gholamreza Anbarjafari. 2020. Human activity recognition-based path planning for autonomous vehicles. *Signal, Image and Video Processing* (2020), 1–8.

[14] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

[15] Jun Wan, Chi Lin, Longyin Wen, Yunan Li, Qiguang Miao, Sergio Escalera, Gholamreza Anbarjafari, Isabelle Guyon, Guodong Guo, and Stan Z Li. 2020. ChaLearn Looking at People: IsoGD and ConGD Large-Scale RGB-D Gesture Recognition. *IEEE Transactions on Cybernetics* (2020).

[16] Qingjiu Zhang and Shiliang Sun. 2010. A centroid k-nearest neighbor method. In *International Conference on Advanced Data Mining and Applications*. Springer, 278–285.