

Automated Highlight Generation of Amateur Table Tennis Videos Using Machine Learning

Vishal Sivakumar¹

Received August 8, 2026

Accepted September 28, 2026

Electronic access October 15, 2026

The widespread use of smartphones has made it common for amateur players to record table tennis matches. However, active gameplay can account for less than half of the recording time (about 40% in our annotated match recordings), making manual review and highlight creation time-consuming. This study describes a machine learning system that automatically generates highlights from amateur table tennis videos by classifying video segments as Action or No Action. Using Temporal Segment Network (TSN) with a ResNet-50 backbone, the model classifies video clips using information from multiple sampled frames. The model does not explicitly perform ball or player tracking or pose detection. We converted a dataset of 28 manually annotated match videos containing over 253,000 labeled frames into over 5,800 fixed-length clips. We split the clips at the video level into training, validation, and test sets. The primary 64-frame model attained 88.7% validation accuracy and 92.28% test accuracy. A temporal sensitivity study showed that the 16-frame configuration achieved 90.10% validation accuracy with a Macro F1 score of 0.8997. The inference time also decreased from 16.55 ms/clip to 5.56 ms/clip compared with the 64-frame model. Another study comparing 10 different train-validation splits across the video set showed consistent performance among combinations. In practice, the generated highlights video reduced total viewing time by about 60–70%. This shows that a machine learning-based approach to automated highlight generation can make reviewing amateur table tennis matches significantly more efficient.

Keywords: Table Tennis, Action Recognition, Video Highlight Generation, Temporal Segment Networks (TSN), Deep Learning, Sports Video Analysis

Introduction

Over the past decade, the extensive adoption of smartphones has made recording amateur sports easier and more common. Athletes and coaches commonly rely on recorded footage to review and analyze match play. Indoor sports like table tennis generate a lot of video content because of frequent match play. However, this excess presents a practical challenge: only a small portion is reviewed, and it can be time-consuming. Also, active match play accounted for around 40% of the annotated footage in our dataset, while the remaining footage is non-action. As a result, much of the recorded content remains underutilized.

This underutilized match footage contains valuable information that can help players understand their weaknesses and improve their game. However, because of excess No Action content, analysis becomes time-consuming and tedious. In addition, these videos can take up lots of storage space. An action-only version of the recording, which is the highlights, can make a big difference. By trimming the video, players can analyze footage more efficiently and save storage space.

Before advances in machine learning and AI, highlight gen-

eration was difficult. The only option was to manually cut out inactivity in the video, a doable but tedious task. Manual methods cannot keep up with the volume of content generated today, which limits the usefulness of these recordings. Also, the large storage space these videos take up is a challenge.

With advances in machine learning and AI, a model can be trained to distinguish patterns throughout video frames and identify different segments. The result is a model that can automatically create highlights from raw video, saving storage and time when analyzing matches. Current approaches to sports video analysis rely on object tracking¹ (e.g., ball, paddle, table), posture classification², or player movements³. These require matches to be recorded so the objects are clearly visible. But in actual amateur recordings, this is not always the case because of camera motion, background clutter, and varying recording angles. Traditional computer vision techniques are less effective in these kinds of settings.

To address this problem, we built a system to analyze amateur table tennis match videos. It identifies segments of active play (“Action”) and periods of inactivity (“No Action”) by working directly with RGB frames. It does not rely on explicit tracking or detection of the ball, paddle, table, or players. For model training and testing, we created and labeled a dataset

¹ Saint Francis High School, Mountain View, CA, USA

using real amateur match recordings. Our dataset reflects typical background activity, camera positions, and recording conditions found in amateur match environments.

Related Work

Automated sports video analysis requires models to obtain relevant information at the frame level. In quick-paced racket sports like table tennis, high-velocity ball trajectories, sudden player movements, and visual clutter complicate this task. Much related work has focused on object tracking, pose estimation, human behavior analysis, action recognition, and action detection.

Roman Voeikov et al. (2020)⁴ developed a real-time model that analyzes table tennis videos to do automatic scoring. Using a multi-task neural network, they detected game events, tracked the ball, and segmented objects including the table and the players. They trained a neural network on an OpenTTGames dataset of table tennis videos at 120 frames per second. Their model attained 97% accuracy in spotting game events and 97.5% in ball detection, with only a 2-pixel RMSE error. The model processed videos in under 6 milliseconds per frame. Results show the model can detect game information to automate scoring, helping referees and coaches while lowering the necessity for manual annotations. However, the system was designed only for ball tracking and referee assistance, not highlight generation. Most importantly, the model was evaluated on its own OpenTTGames dataset, so it was not validated on other camera angles or conditions, making it unclear whether it can be used in real-life situations where backgrounds may vary.

Jiang Bian et al. (2024)⁵ aimed to advance action recognition and localization research by providing fully annotated realistic match footage for 14 table tennis action classes. Rather than a detection model, they evaluated several advanced action recognition models, including TSM, TSN, SlowFast, Video Swing Transformer, VideoMAE V2, BSN, BMN, and TCANet+. VideoMAE V2 achieved 85% accuracy on an 8-class task, but existing localization models reached only 48% AUC. The results show that action detection in table tennis remains difficult, and even advanced models have difficulty because of the ball's speed and small object size. The limitations indicate that present models achieve decent accuracy because of high action density, short action duration, fast ball speed, and limited video resolution. The authors explain that more advanced strategies are needed to attain more accurate results in fast sports like table tennis.

Several of the models Bian et al. evaluated represent important developments in video action recognition. TSM⁶ adds temporal information to 2D CNN models by shifting features over neighboring frames, while SlowFast⁷ Networks use separate pathways to capture slower spatial information and faster

motion information. More recent approaches such as Video Swin Transformer⁸ and VideoMAE V2⁹ use Transformer-based and self-supervised learning techniques for video understanding. These methods extract temporal information from video rather than relying only on individual images.

James Hong et al. (2022)¹⁰ focused on finding exact frames where key events happen in sports videos, such as hitting a tennis ball or a figure skater landing. Their E2E-Spot model learned spatial and temporal features from video data. In tests on many sports datasets, E2E-Spot outperformed earlier models in action detection and temporal segmentation. For example, on the SoccerNet-v2 dataset, it achieved an average mAP of 74.05% and a tight average mAP of 61.82%. The model also improved mean precision by 4 to 11 percentage points over different datasets. Their findings show that learning spatial and temporal information can be more important than detecting actions, and they released additional annotations to support future work in sports event detection.

Humam Alwassel et al. (2018)¹¹ aimed to find action portions in long videos by only using relevant sections rather than the entire video. They used an LSTM-based RNN to identify which parts of the video to process. The model attained 30.8% mean precision on THUMOS14 while viewing only 17.3% of the video, showing that selective processing for long videos can work. However, the system finds approximate areas where actions may occur rather than generating continuous highlight segments. It was also evaluated on general datasets rather than sports-specific datasets.

Another related area is temporal action localization, where models identify an action and determine where it starts and ends within an untrimmed video. BSN¹² introduced a boundary-sensitive approach to generate temporal activity proposals, and BMN¹³ later modeled relationships between possible start and end boundaries. More recent approaches such as ActionFormer¹⁴ and TriDet¹⁵ have also focused on improving action and boundary localization. These approaches are relevant to sports highlight generation because identifying an activity's start and end is important when converting a continuous recording into meaningful segments.

Valli Nayagam et al. (2026)¹⁶ used a Convolutional Neural Network (CNN) and Transformer to generate highlights from football and cricket videos. For football they reported 98-99% accuracy and for Cricket the accuracy was 95-97%. But they explained that their performance was affected by amount of training data and camera angle. They suggested extending the approach to other sports and adding ball and player tracking in the future.

Zhongjie Wang et al. (2025)¹⁷ study focused on recognizing postures for video analysis in sports such as shooting, weightlifting, running, and pole vaulting. They reported 97.6% accuracy, though background objects were sometimes misidentified as athletes.

Kulkarni and Shenoy (2021)¹⁸ focused on recognizing and classifying specific strokes using controlled datasets. Authors Kaustubh Milind Kulkarni and Sucheth Shenoy collected a large dataset with over 22,000 stroke videos from 14 professional players and developed a pose-based Temporal CNN to recognize 11 table tennis strokes. In their approach, the authors relied on human pose estimation to reduce the impact of background changes and make the model work well with different players. The system reached a test accuracy of 99.37%, and it also correctly classified strokes for a player it had not seen before with 98.72% accuracy. These results suggest that focusing on player poses and movement patterns can help recognize specific actions. However, this data was collected in a controlled environment, including a camera placed at the center of the table on the net. There was a clear, consistent, and unobstructed view of player movements. The work also focuses on stroke clips captured in a controlled setup rather than continuous match analysis or highlight generation.

Other recent table tennis studies have approached video analysis in different ways. Adaptive Temporal Aggregation¹⁹ has been used for table tennis shot recognition under changes in players and viewpoints, while Dong and Yan²⁰ combined visual features with a Transformer for recognizing table tennis player actions, including a NoAction category. Song et al.²¹ used pose estimation and spatial-temporal graph networks for technical and tactical action recognition across multiple players. Wei and Chang²² studied automatic segmentation of table tennis match videos based on player actions, particularly for separating useful match content from longer recordings.

Researchers have also studied similar problems in other racket sports. TenniSet²³ provides densely labeled tennis match videos for fine-grained event recognition and temporal localization. In badminton, TemPose²⁴ uses skeleton-based temporal representations to recognize fine-grained player motions. These studies show that racket-sport video analysis can benefit from temporal information. Still, many approaches focus on specific strokes, events, poses, or player movements instead of directly producing Action/No Action highlights from ordinary amateur recordings. Automated sports video analysis requires models to obtain relevant information at the frame level. In fast racket sports like table tennis, high-velocity ball trajectories, sudden player movements, and visual clutter complicate this task. Much related work has focused on object tracking, pose estimation, human behavior analysis, and action detection.

Research Gap and Contribution

Most studies²⁵ so far have focused on specific problems such as stroke classification, ball tracking, or event spotting²⁶ using some form of specialized test settings. Some reported high accuracies (90+%), but these results were often obtained using

trimmed clips, high-frame-rate videos, or specialized camera setups that provided clearer views of the players and gameplay. Relatively limited work has focused on automatically generating highlights from untrimmed amateur table tennis videos recorded using smartphones at standard frame rates such as 30 fps. Unlike several existing approaches, this study does not explicitly perform ball tracking or pose estimation. The approach also does not require specialized camera placement and is designed for single-camera smartphone recordings from outside the playing area.

The contributions of this work can be expressed in three parts. First is collecting and annotating real-life amateur match videos with frames labeled as Action or No Action. These videos have a variety of backgrounds (e.g., matches happening next to the match being recorded, different color backgrounds, big venues, small venues, etc) and recording conditions representative of the dataset used in this study. Second is a model trained using labeled video clips to classify Action and No Action without explicitly tracking the ball, paddle, table, or players, or using pose estimation. It does not rely on object tracking or pose estimation, since the ball may not always be visible in amateur recordings. Third is an inference and post-processing pipeline that gathers and smooths model predictions into action segments that are put together for highlight generation.

Dataset

We used a data-centric approach to build a dataset that represents real-world situations. The dataset uses the variability found in amateur recordings rather than using controlled data. The final dataset includes 28 videos of the author's actual matches, recorded between 2022 and 2026 at different competitive events using multiple smartphone and tablet devices. The recording device was mounted on a tripod. The videos cover a wide range of recording conditions, including variations in background, lighting, venue, opponents, dates, devices, and camera position, as shown in Figure 1. Some videos have simpler settings with less background variation, while others are more complex, with crowd movement, multiple tables, and changing visual conditions. Some tournament recordings also contain other active table tennis matches in the background, providing more challenging multi-table scenes.

We manually annotated all videos using a binary labeling scheme in CVAT 2.4, and Figure 2 illustrates this with a sample video stream. For each video, frames were annotated as "Action" or "No Action," and a metadata file was generated. Action frames depict active gameplay during points, such as serving and rallies. No Action frames include pauses between points, such as getting the ball or thinking before a point. It can also include timeouts or breaks between games. Player movements occurring between points, such as walking, re-

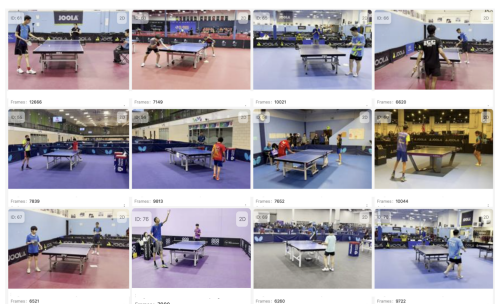


Figure. 1 Sample frames from the dataset showing variation in background, lighting conditions, and camera positioning across recordings. Thumbnails from 12 input videos.



Figure. 2 Manual annotation of Action and No Action segments on the sample video timeline.

trieving the ball, or preparing for the next point, were labeled as No Action rather than Action. Between every Action segment is a No Action segment.

Across the 28 videos, the total annotated duration was around 8,440.7 seconds (140.68 minutes). We manually annotated each video at the frame level into two classes: Action (periods of active gameplay) and No Action (inactivity). Active gameplay accounted for around 38.78% of the annotated footage. The remaining 61.22% consisted of non-action periods, showing the amount of time outside active gameplay and motivating the need for automatic highlight generation. The annotated dataset contains 253,150 labeled frames across all videos. We calculated duration using each source video's measured FPS. We split the 28 annotated videos into 20 training videos, 4 validation videos, and 4 test videos. Table 1 summarizes the 24 training and validation videos. We kept the four test videos outside of training and validation and did not use them for training or selection. Table 2 summarizes them separately.

Methodology

Data Preprocessing

From the annotated videos described above, we generated training clips. Each clip was 64 frames at 30 fps, or about

2.13 seconds. We chose this frame size because it captures enough meaningful sequence without being too long. In table tennis, a rally can be about 2 to 15 seconds long, and there are more No Action frames than Action frames in a normal match recording. To balance the training dataset, we use a stride-based strategy to generate the clips. For Action clips, we used a fixed stride of 32 frames, which is 50% overlap across clips. For No Action clips, we used an adaptive stride of 32, 48, or 64 frames, depending on the length of the non-action segments. The adaptive No Action stride reduced highly overlapping and redundant clips while retaining denser sampling of shorter Action periods. Overall, this process produced 5,847 clips from the 28 annotated videos. All clips contained 64 consecutive frames.

Train-Validation-Test Split

We split the dataset at the video level to avoid clip leakage, using 20 videos for training and 4 videos for validation (Input013, Input031, Input052, and Input076). We used an additional four videos (Input016, Input075, Input078, and Input080) as the test set. Table 3 shows the distribution of Action and No Action clips across the three sets. This improved the source-frame Action/No Action imbalance from 1:1.58 to 1:1.09 in the generated clips.

To measure model effectiveness over various validation video groups, we conducted a sensitivity study with 10 different train-validation splits (other parameters kept the same). The main difference between groups was that, within the dataset's 24 training-validation videos, each group used a different set of 4 validation videos. We limited this sensitivity study to 10 splits because of limited compute resources.

We performed all clip splits at the whole-video level. This ensures that clips from same source video were not included in both the training and validation sets. We followed this rule for each of the 10 groups reported in Table 7. We agree, however, that a whole-video split does not fully account for similarities in players, venues, recording conditions, or camera positions across videos.

Frame-Differencing Baseline

For baseline comparison, we tested with a frame-differencing methodology. Frame differencing is a commonly used motion-detection approach that measures changes between consecutive video frames.^{27,28} For this, we converted consecutive video frames to grayscale and calculated the mean absolute pixel difference as a motion score. We smoothed the scores with a 5-frame moving average and classified frames above a single threshold as Action. We chose the threshold using training videos by maximizing Macro F1, which resulted in a threshold of 2.551.

Table 1 List of annotated videos for Training-Validation.

Video	Action Frames	No Action Frames	Action Duration (s)	No Action Duration (s)	Total Annotated Duration (s)
Input001.MOV	3,657	3,761	121.962	125.431	247.393
Input002.MOV	3,155	5,608	105.220	187.029	292.249
Input003.MOV	3,248	6,098	108.322	203.371	311.693
Input004.MOV	3,727	4,020	124.296	134.068	258.364
Input005.MOV	3,951	7,686	131.814	256.423	388.237
Input007.MOV	4,024	3,363	134.250	112.197	246.447
Input009.MOV	3,502	5,220	116.835	174.151	290.986
Input010.MOV	3,432	6,070	114.500	202.509	317.009
Input013.MOV	3,413	5,176	113.778	172.550	286.328
Input018.MOV	4,845	6,707	161.490	223.553	385.043
Input020.MOV	3,361	3,949	112.026	131.625	243.651
Input030.MOV	2,732	5,065	91.146	168.980	260.126
Input031.MOV	3,953	5,823	131.881	194.269	326.15
Input035.MOV	3,178	4,412	105.971	147.119	253.09
Input037.MOV	4,104	5,885	136.919	196.338	333.257
Input041.MOV	3,120	3,467	103.994	115.560	219.554
Input049.MOV	2,761	3,725	92.028	124.159	216.187
Input052.MOV	4,110	8,514	136.992	283.783	420.775
Input061.MOV	4,819	6,458	160.623	215.253	375.876
Input064.MOV	2,683	3,541	89.428	118.026	207.454
Input069.MOV	3,078	4,037	102.594	134.558	237.152
Input071.MOV	3,279	6,712	109.293	223.720	333.013
Input073.MOV	3,415	6,270	113.826	208.987	322.813
Input076.MOV	3,402	8,661	113.393	288.682	402.075
Total (24)	84,949	130,228	2,832.58	4,342.34	7,174.92

Table 2 List of annotated videos for Test

Video	Action Frames	No Action Frames	Action Duration (s)	No Action Duration (s)	Total Annotated Duration (s)
Input016.MOV	3,208	6,591	106.944	219.722	326.667
Input075.MOV	2,929	4,187	97.627	139.558	237.185
Input078.MOV	3,727	9,388	124.226	312.914	437.140
Input080.MOV	3,347	4,596	111.560	153.190	264.750
Total (4)	13,211	24,762	440.36	825.38	1,265.74

Model Architecture and Training

The model is based on a Temporal Segment Network (TSN)²⁹ implemented in MMAAction2³⁰ as a Recognizer2D, with a ResNet-50³¹ backbone pretrained on ImageNet. TSN processes multiple frames from a short clip, classifies each frame, and averages the predictions to determine whether the clip contains Action or No Action. In our implementation, each input clip is 64 frames long (~ 2.13 seconds at 30 fps, the standard frame rate for smartphones). TSNHead performs the final classification by combining the predictions from the 64 sampled frames to produce a single prediction for the clip. The model outputs probabilities for two classes: Action and No Action. We applied a dropout layer with rate of 0.5 before the final classifier to reduce overfitting. The model uses Softmax Action and No Action classification. Below is the training pipeline sequence:

Clip $\rightarrow$ *Frame-Sampling* $\rightarrow$ *Decode* $\rightarrow$ *Resize* $\rightarrow$ *MultiScale-Crop* $\rightarrow$ *Flip* $\rightarrow$ *TSN+ResNet50* $\rightarrow$ *Model*

Within each clip, TSN uses clip_len = 1 (one frame at a time), num_clips = 64, and frame_interval = 1 (consecutive frames) to sample the frames. The model therefore processes 64 consecutive frames independently through the ResNet-50 backbone (2D convolutions). Source videos are usually recorded on mobile devices at 1920×1080. ResNet-50 accepts a square crop (here 224×224); several preprocessing steps are applied before the frames are passed to the model. The steps are as follows: Step 1 decodes the frames from the clips. Step 2 resizes the short side to 256 while keeping the aspect ratio (e.g., 1920×1080 to approximately 455×256). Step 3 applies MultiScaleCrop (MSC) to select a 224-scale crop region. Step 4 resizes the crop to 224×224. Step 5 is a random horizontal flip with probability 0.5 to improve variability.

MultiScaleCrop³² in Step 3 is important for making the

Table 3 Distribution of clips across training, validation and test sets

Dataset Split	Videos	Total Clips	Action Clips	No Action Clips	Action %	No Action %
Training	20	4,089	1,997	2,092	48.8%	51.2%
Validation	4	929	426	503	45.9%	54.1%
Training + Validation	24	5,018	2,423	2,595	48.3%	51.7%
Test	4	829	372	457	44.9%	55.1%
Total	28	5,847	2,795	3,052	47.8%	52.2%

model reliable and stable. We ran many experiments to evaluate the best setting. With `random_crop=False`, crops are taken from one of the five positions (center, top-left, top-right, bottom-left, bottom-right), using mild scales (1.0, 0.875) and `max_wh_scale_gap=1`. Across all epochs, the model can now see different spatial contexts, such as players, the table, and outer areas, instead of always training on the center crop. Figure 3 shows the sample frame with MSC selection for the five positions. This shows how one frame can give multiple training views and help the model learn better even with the forced square crop.

We trained the model using stochastic gradient descent (SGD) with the following settings: initial learning rate of 0.01, momentum at 0.9, weight decay of 1×10^{-4} . A cosine annealing learning-rate schedule gradually reduced the learning rate over up to 150 training epochs. We trained with a batch size of 4 and evaluated validation accuracy after every epoch. We used early stopping after at least 120 epochs, with a patience of 20 epochs. In most experiments, convergence occurred between checkpoints 90 and 105. We selected the final model based on validation scores at the checkpoint. To study the effect of temporal sampling, we repeated the entire training procedure using four sampling configurations (64/int1, 32/int2, 16/int4, and 8/int8). All experiments used the same architecture, preprocessing pipeline, optimizer, and learning-rate schedule; only the temporal sampling rate within each 64-frame window was changed.

The following sequence represents the inference and validation pipeline:

Clip→**Frame-Sampling**→**Decode**→**Resize**→**CenterCrop**→**Model**→**Probability output**

Validation and inference follow a similar pipeline to training, except they use a deterministic 224x224 CenterCrop and no Flip before passing to the model. MSC was mainly needed during training for the model to learn input variations.

Model Inference and Post-processing

The following is the highlight generation flow. Inference flow is described above.

Video→**Sliding-Window**→**Inference**→**Probability output**→**Smoothen**→**ffmpeg**→**Highlights**

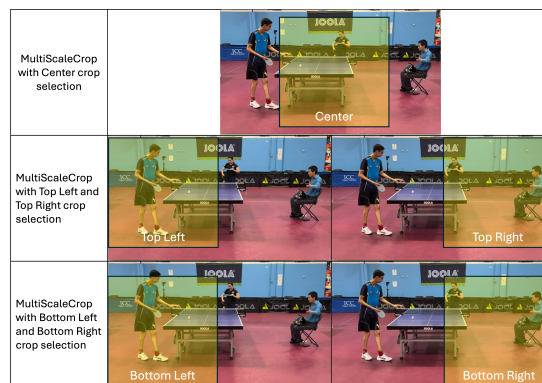


Figure. 3 MultiScaleCrop with five possible crop selections for a given sample frame is shown (Center, Top Left, Top Right, Bottom Left, Bottom Right)

For full-length videos, inference uses a sliding-window with a window of 64 frames and a stride of 8, so consecutive windows overlap by around 87.5%. Figure 4 shows the overlapping clips.

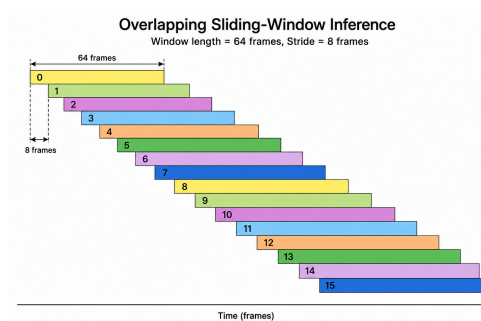


Figure. 4 Inference strategy - Overlapping sliding windows (64 frames, stride 8) used

We build 64-frame windows and pass them through the model to predict actions. For the next window, we shift by eight frames, creating significant overlap to produce consistent predictions throughout the video. For each window, the trained TSN model outputs action probabilities. Because frames are in multiple windows, each frame gets many predic-

tions. We combine these by averaging the Action probabilities from all windows that cover that frame. That yields a continuous probability. Preprocessing during inference matches validation: we resize the short side to 256, then CenterCrop to 224×224.

During post-processing, we then convert the frame-level probability into highlight segments. The probability threshold is 0.5, so frames at or above this value are labeled Action. Short No Action gaps of 0.3 seconds or less are merged, and Action segments shorter than 0.5 seconds are removed. Finally, each Action segment is extended by 0.15 seconds at the end to produce the segment. This 0.15 s extension corrects boundary errors in the order of half the 8-frame inference stride, intended to offset small early cutoffs after thresholding rather than add trailing highlight padding. If a larger inference stride (eg 16 or 32) were used, this constant should be revisited. We trim the final Action segments from the video and concatenate them in chronological order to form a highlight reel. This preserves the main rallies and shortens long match recordings. For full-video evaluations, we calculated frame-level accuracy after the temporal post-processing steps described above. We calculated clip-level validation and test metrics directly from the model predictions, without temporal post-processing.

Model Evaluation Metrics

To measure model effectiveness, we used several tests that measure different aspects. At the clip level, we report standard classification metrics (Accuracy, Precision, Recall, F1, and PR-AUC) for Action and No Action on the validation and test clips. We generated classification metrics using the selected model checkpoint. We also performed temporal sensitivity analysis, sliding-window stride analysis, a 10-group train-validation sensitivity study, and Grad-CAM saliency map analysis to examine model effectiveness and behavior within different settings.

Results

This section summarizes the results of the study. First is the 64-frame model's training and validation accuracy over epochs, followed by clip-level evaluation on the four validation videos and four test videos. We then compare the model with the frame-differencing baseline and evaluate temporal sampling, different train-validation splits, and sliding-window stride. Finally, Grad-CAM analysis and model efficiency are presented.

All training, validation, and full-video inference experiments were run on a Ubuntu Linux 22.04.5 LTS server with dual Intel Xeon Gold 6448H CPUs (128 threads), 1 TB system RAM, and two NVIDIA H100 PCIe GPUs (80 GB

each; NVIDIA driver 570.195, CUDA 12.8). The software stack was Python 3.10, PyTorch 2.8 (CUDA 12.8 builds), and MMAAction2 1.2 / MMEEngine 0.10 for TSN training and validation, with Decord for video decoding and FFmpeg for highlight export. During training, 4 clips were run in parallel, while validation and inference was done one by one.

Training Results with 64 Frame Model

Figure 5 shows the model training and validation accuracy over the epochs. During the early epochs, the model remained close to chance-level performance, with validation accuracy staying around 50–54%. After approximately Epochs 50–60, the loss began to decrease more noticeably, coinciding with an increase in classification accuracy. At later epochs, training accuracy continued to increase while validation accuracy showed much smaller improvement, indicating some overfitting. Dropout, data augmentation, weight decay, and early stopping were used to reduce overfitting during training. Epoch 95 was used for the following 64-frame clip-level evaluation.

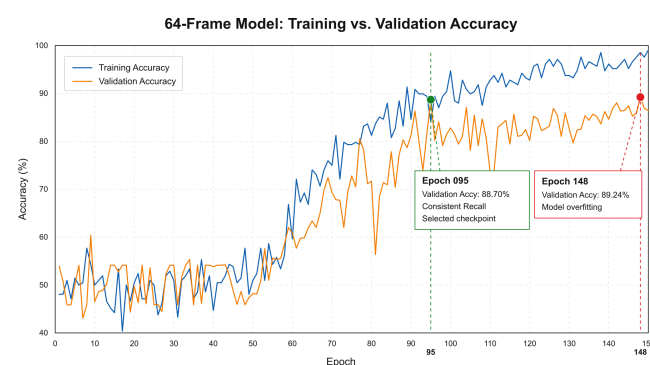


Figure. 5 Train and Validation accuracy vs epoch with 64-frame model

Table 4 summarizes clip-level evaluation metrics on the four validation videos at epoch-95. Overall accuracy was consistently strong, ranging from about 87.8% to 89.5%. Action and No Action F1 scores were fairly well balanced (roughly 86.7–90.3%), and PR-AUC was high for both classes (about 0.93–0.98).

Input076 shows a noticeable difference between Action precision (79.83%) and recall (97.94%). This means that most true Action clips were detected, but some No Action clips were classified as Action. Input076 was recorded in a more complex tournament setting where several other table tennis matches are visible around the foreground match. Since the model classifies the full RGB frame, activity from neighboring tables may contribute to some false Action predictions. Figure 6 shows two different frames from this recording. The

Table 4 Summary of evaluation metrics for videos in the validation set with 64-frame model

Clip Name	Label	Accuracy	Precision	Recall	F1	PR-AUC
Input013	Action	88.11%	91.21%	85.57%	88.30%	0.964
	No Action		85.11%	90.91%	87.91%	0.935
Input031	Action	89.54%	90.09%	87.72%	88.89%	0.947
	No Action		89.06%	91.20%	90.12%	0.965
Input052	Action	87.83%	85.25%	88.14%	86.67%	0.939
	No Action		90.07%	87.59%	88.81%	0.963
Input076	Action	89.26%	79.83%	97.94%	87.96%	0.928
	No Action		98.37%	83.45%	90.30%	0.980

frame on the left shows a player with a towel (No Action) but a player from an adjacent table in Action. (False Positive example) Frame on the right shows three players in Action at the same time but from two different matches demonstrating a complex visual.

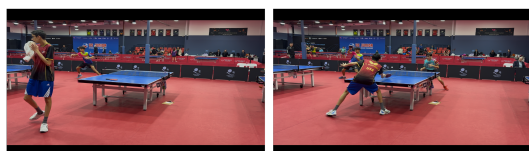


Figure. 6 Example frames from Input076 showing the complex tournament setting, with activity from multiple neighboring tables visible in the full RGB frame.

Figure 7 shows Action and No Action F1 side by side for each validation video.

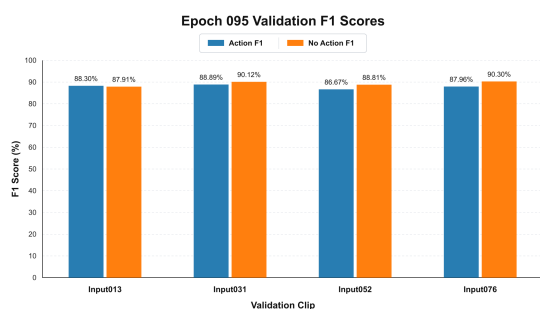


Figure. 7 F1 scores to predict Action and No Action in validation videos with 64-frame model

On all four clips, the two bars are close in height, showing that the model performs well on both classes. Input031 has the highest pair of F1 scores (Action 88.9%, No Action 90.1%). Input013 is quite close (Action 88.3%, No Action 87.9%). In-

put052 is a bit lower on Action F1 (86.7%) than No Action (88.8%). Input076 keeps a solid Action F1 (88.0%), despite the tradeoff between precision and recall, and No Action F1 is still high (90.3%).

Baseline Comparison

A simple frame-differencing baseline was evaluated on the four validation videos to determine whether pixel-level motion alone could distinguish Action from No Action. Table 5 compares the baseline with the TSN-based approach using the same manually labeled frames. The frame-differencing baseline achieved 55.9% frame accuracy and a Macro F1 score of 0.49, compared with 88.7% and 0.8865 for the TSN-based approach.

Table 5 Comparison vs Baseline

Method	Frame Accuracy	Macro F1
Frame Differencing	55.9%	0.49
TSN-based Approach	88.7%	0.8865

This shows that pixel-level motion alone does not reliably separate active gameplay from other movement in the videos. Movements between points and activity elsewhere in the scene can also produce substantial frame differences even when the foreground match is labeled No Action.

Sensitivity of Temporal Sampling

A temporal sensitivity study was conducted to see the impact of using different sampling intervals on model performance. Table 6 below summarizes the details. Among the four models, the 16/int4 model achieved the highest validation accuracy (90.10%) and Macro F1 score (0.8997). It also required much less training and inference time than the 64/int1 model. This shows that even sampling every fourth frame (16/int4) can preserve important information while reducing computational cost by a lot. This provides an effective balance between accuracy and efficiency.

Sensitivity of Train-Validation Split

To test the effectiveness of the training method, we evaluated ten variations of train-validation mix. In each variation, we used different combinations of input videos. (Labelled as Group 1 to 10 in Table 7) This sensitivity study was conducted on the 16 frame model (16/Int4). Experimentation was limited to 10 variations due to server resource constraints. Table 7 summarizes the results. Across all validation groups, model accuracy ranged from 85.92% to 93.95%. Both Action and No

Table 6 Temporal sensitivity study of the model on validation scores, training and inference time

Configuration	Temporal Sampling	Validation Accuracy	Validation Macro F1	Training Time (h)	Inference Time (ms/clip)
64/int1	64 clips $\times$ interval 1	88.70%	0.8865	30.4	16.55
32/int2	32 clips $\times$ interval 2	85.15%	0.8508	18.5	9.10
16/int4	16 clips $\times$ interval 4	90.10%	0.8997	17.8	5.56
8/int8	8 clips $\times$ interval 8	88.05%	0.8794	16.4	5.80

Action classes achieved high precision, recall, F1 score, and PR-AUC values across most groups.

Table 7 Evaluation metrics across 10 different train-validation groups

Group	Type	Accuracy	Precision	Recall	F1	PR-AUC
Group 1	Action	90.10%	0.9196	0.8592	88.83%	0.9498
	No Action		0.8870	0.9364	91.10%	0.9724
Group 2	Action	85.92%	0.8645	0.8429	85.35%	0.9087
	No Action		0.8545	0.8747	86.45%	0.9489
Group 3	Action	93.88%	0.9532	0.9178	93.51%	0.9800
	No Action		0.9264	0.9582	94.20%	0.9822
Group 4	Action	87.88%	0.8489	0.8952	87.14%	0.9330
	No Action		0.9068	0.8648	88.53%	0.9584
Group 5	Action	87.86%	0.9147	0.8265	86.84%	0.9466
	No Action		0.8506	0.9276	88.74%	0.9533
Group 6	Action	86.21%	0.8154	0.9257	86.71%	0.9090
	No Action		0.9195	0.8020	85.68%	0.9273
Group 7	Action	91.39%	0.8832	0.9380	90.97%	0.9625
	No Action		0.9436	0.8932	91.77%	0.9781
Group 8	Action	93.95%	0.9306	0.9499	94.01%	0.9796
	No Action		0.9488	0.9292	93.89%	0.9844
Group 9	Action	90.22%	0.8834	0.9244	90.35%	0.9587
	No Action		0.9223	0.8804	90.09%	0.9561
Group 10	Action	90.76%	0.8866	0.9207	90.33%	0.9442
	No Action		0.9275	0.8960	91.14%	0.9666

Some variation between classes was observed across the different groups. For example, Group 6 achieved a higher Action recall of 92.57% and a lower No Action recall of 80.20%. During training, validation performance showed noticeable fluctuations across checkpoints. When different checkpoints were examined, improvements on some validation videos were sometimes accompanied by lower performance on others. This suggests that the model was somewhat sensitive to checkpoint selection and did not achieve equally

stable performance across all validation videos, which may help explain the lower No Action recall observed in Group 6.

Figure 8 below plots the F1 scores for each of the ten validation groups for Action and No Action. Across the 10 groups, Action achieved a mean F1 score of 89.40% with a standard deviation of 4.33%, while No Action achieved a mean F1 score of 90.16% with a standard deviation of 4.26%. Overall, F1 scores remained relatively consistent across the different train-validation splits.

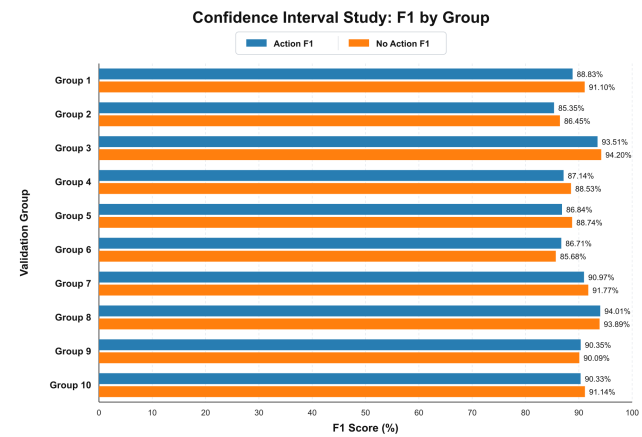
**Figure. 8** F1 scores to predict Action and No Action in 10 different train-validation groups.

Figure 9 plots the Recall scores for each validation group for Action and No Action.

The Action class achieved a mean Recall score of 90.00% $\pm$ 6.17%, while the No Action class achieved a mean Recall score of 89.63% $\pm$ 7.81%. Although recall had more variation than the F1 score, the mean recall remained close to 90% for both classes across the 10 train-validation splits.

Sensitivity of Sliding-Window Stride

A sensitivity study was also conducted to evaluate the effect of the sliding-window stride used during full-video inference. The original inference pipeline used a 64-frame window with a stride of 8 frames, resulting in 87.5% overlap between consecutive windows. We compared this with strides of 16 and 32 frames, corresponding to 75% and 50% overlap. Table 8



Figure. 9 Recall scores to predict Action and No Action in 10 different train-validation groups. Each group represents a different train-validation split of videos.

summarizes the results across the four validation videos. End-Boundary Error measures how many frames the predicted end of an Action segment differs from the manually labeled actual end.

Table 8 Sensitivity of sliding-window stride during full-video inference

Stride	Overlap	Frame Accuracy	Median End-Boundary Error by Video (frames)
8	87.5%	86.97%	6, 14, 26, 10 (Avg: 14.0)
16	75.0%	86.87%	10, 13, 26.5, 11 (Avg: 15.1)
32	50.0%	86.98%	12, 16, 34, 14 (Avg: 19.0)

Frame accuracy was nearly identical across the three stride settings. However, increasing the stride resulted in larger Action end-boundary errors, with the average median error increasing from 14.0 frames at stride 8 to 19.0 frames at stride 32. Since accurate segment boundaries are important for highlight generation, stride 8 was retained to provide denser temporal predictions and better end-boundary localization.

Test Set Results

The selected model was evaluated on the four annotated test videos, which were not used for model training or model selection. Table 9 shows the results. Across the four test videos, the model achieved a pooled accuracy of 92.28% and Macro F1 score of 0.9221. Input016 achieved the highest accuracy at 96.50%, while Input078 had the lowest accuracy at 88.93%. Input078 was recorded in a tournament environment with mul-

tle active table tennis matches visible in the background, which may have contributed to the lower performance.

Table 9 Classification results on the test set

Clip Name	Type	Accuracy	Precision	Recall	F1	PR-AUC
Input016	Action	96.50%	96.55%	95.45%	96.00%	0.991
	No		96.46%	97.32%	96.89%	0.993
Input075	Action	92.44%	87.10%	98.78%	92.57%	0.995
	No		98.73%	86.67%	92.31%	0.996
Input078	Action	88.93%	88.12%	83.96%	85.99%	0.953
	No		89.44%	92.31%	90.85%	0.947
Input080	Action	92.31%	92.63%	91.67%	92.15%	0.977
	No		92.00%	92.93%	92.46%	0.984

Model Interpretation

To better understand how the model identifies between Action and No Action, Grad-CAM saliency maps were generated for sample validation clips using the trained 16/Int4 TSN and ResNet-50 model. Grad-CAM produces heatmaps that highlight the regions of each frame contributing most to the model's prediction, where warmer colors indicate greater influence on the classification decision. Grad-CAM was computed at the final ReLU layer of the ResNet-50 backbone (backbone/layer4/2/relu). Channel weights were calculated by spatially averaging the gradients for each TSN segment, and the resulting activation maps were combined and normalized. For visualization, the peak-activation frame was shown along with selected evenly spaced frames from the clip. Example clips were selected from the epoch-148 validation predictions using high-confidence correct Action and No Action predictions, along with the highest-confidence false positive and false negative predictions. Four categories of predictions were examined: correct Action, correct No Action, false positives, and false negatives (Figures 10–13). This qualitative analysis was used to examine the spatial regions that contributed to different types of model predictions. Grad-CAM visualizes spatial regions that contribute to a prediction and does not directly explain temporal behavior. Temporal information in the TSN model is instead incorporated through multi-frame sampling and consensus across the clip.

We did not perform the additional parameter-randomization, label-randomization, or quantitative background-masking experiments in this study, and therefore we have limited our interpretation of Grad-CAM to qualitative visualization rather than using it as evidence of model generalization or background independence.

Figure 10 illustrates correct Action predictions. In these examples, stronger Grad-CAM activation appears around re-

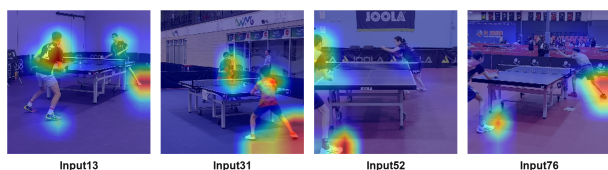


Figure. 10 Correct Action Predictions

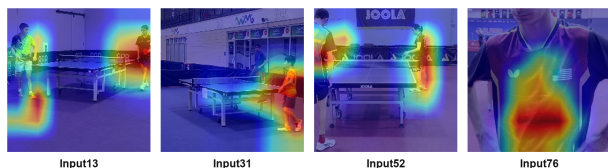


Figure. 11 Correct No Action Prediction

gions containing the players, playing arm, and table during active rallies. These examples suggest that player and table regions contributed to the model's Action predictions.

Figure 11 shows correct No Action predictions. In these examples, the Grad-CAM activation is distributed differently from the active-rally examples, with less concentrated activation around active play over the table. These visualizations provide examples of the spatial regions contributing to correct No Action classifications.

Figure 12 shows false positive predictions. In these examples, Grad-CAM activation appears around the players and playing area even though the clips were labeled No Action. This suggests that player movement during No Action periods can sometimes contribute to an Action prediction.

Figure 13 shows false negative predictions, where clips labeled Action were classified as No Action. The Grad-CAM maps show the spatial regions that contributed to these incorrect predictions. Compared with the correct Action examples, activation in these examples can be less concentrated around the active playing regions.

Model Efficiency

Table 10 below summarizes the performance of the generated highlights with the 64-frame model compared with manual annotations for five match videos. Accuracy was calculated by comparing the model's predictions with manual annotations for every video frame. The table also reports the reductions in video duration and file size. Video viewing time was reduced by 60–70% while preserving the Action segments. Because the highlights were re-encoded using the H.264 codec by ffmpeg, the video sizes were around 85% to 95% smaller than the original recordings. This additional saving above and beyond time reduction is due to ffmpeg compression and is not a con-

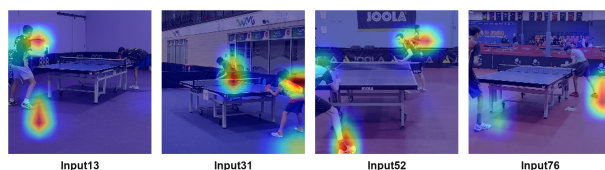


Figure. 12 False Positives Predictions



Figure. 13 False Negatives Predictions

tribution of this study. These results show that the system not only achieves accurate highlight detection but also produces videos that are much easier to store, share, and review.

Discussion

The widespread adoption of smartphones has simplified the recording of amateur table tennis matches, but they are not as useful because they lack automated analysis. This study shows that a machine learning model trained to detect Action and No Action segments can address this problem effectively. Prior approaches have relied on object tracking and pose estimation but could not be applied to amateur recordings. Unlike approaches that explicitly use ball tracking or pose estimation, this method directly classifies video clips from RGB frames without separately detecting or tracking the ball, paddle, table, or players. The model achieved high performance on many metrics, including accuracy, precision, recall, F1, and PR-AUC. The Grad-CAM saliency maps also provided a qualitative view of the spatial regions that influenced different predictions, with many examples showing activation around the players and playing area. These results show that a simpler binary classification approach can identify meaningful gameplay segments in amateur match videos.

The combination of ResNet-50 and Temporal Segment Networks (TSN) worked well to classify Action vs. No Action. ResNet-50 extracted spatial features from individual frames, while TSN combined predictions of the sampled frames to produce a classification for each clip. The temporal sensitivity study also showed that reducing the number of sampled frames could maintain reasonable performance while lowering computational cost. Although this method achieved strong performance, we could not try more designs or models due

Table 10 Model efficiency comparing manual annotation vs generated highlights

Video Name	Frame Detection Accuracy	Original Video (Size, Duration)	Highlight Video (Size, Duration)	Time Reduction	Size Reduction
Input013	88.40%	535 MB, 4:48s	34 MB, 1:51s	61.2%	94%
Input016	92.74%	610 MB, 5:28s	41 MB, 1:55s	64.9%	93%
Input018	91.03%	422 MB, 6:27s	61 MB, 2:40s	58.7%	86%
Input052	86.41%	462 MB, 7:02s	59 MB, 2:46s	60.7%	87%
Input078	90.03%	479 MB, 7:19s	55 MB, 2:12s	70.0%	89%

to limited computing resources. Therefore, we did not test more recent models like video transformers or hybrid CNN-transformers. We also did not perform extensive parameter tuning as we were limited to a practical number of experiments. Another opportunity for future work is investigating more efficient training and inference approaches that reduce the need for high-end server platforms.

The system offers value for players and coaches beyond just classification performance. The generated highlight videos substantially reduced the amount of match footage that needed to be viewed. Across the evaluated videos, highlight duration was reduced by approximately 59–70%. The exported highlight files were also considerably smaller than the original recordings. Any file-size reduction beyond the duration reduction is due to re-encoding the generated highlights with FFmpeg, so it cannot be attributed solely to the highlight-generation method. The reduction in viewing time more directly demonstrates the usefulness of automatically identifying the active portions of a match.

This work also supports the growing area of sports video analytics by dealing with a different problem than much previous table tennis research. Current methods commonly focus on stroke recognition, player pose estimation, or ball tracking, often using carefully controlled datasets or professional recordings. In comparison, this study prioritizes automatically identifying active gameplay in untrimmed amateur match videos recorded under different real-life conditions. This provides a practical way to shorten long match recordings and make them easier for players and coaches to review.

Limitations

Even though the results are promising, there are still some limitations. First, the dataset is small, comprising 28 match videos—24 used for training and validation and four set aside for testing. We divided the data cleanly at the entire-video level to prevent leakage during training. Nevertheless, we did not split the dataset by player, venue, or recording session, and the test set contained only four videos. We have identified this as an area for future work.

We trained the model on videos recorded with a stationary smartphone, a common setup among recreational and compet-

itive table tennis players. As a result, all recordings had a constant resolution and frame rate, although they were captured at different times and venues. Although this choice agrees with the intent, the model was not evaluated on other camera angles (e.g., bird’s-eye, side view, handheld camera view) used in professional table tennis environments. Future work can train and evaluate the model using various camera angles, perspectives, resolutions, and frame speeds to improve generalization across several recording conditions.

Another limitation in the model development was computing power. We evaluated only a finite number of training configurations due to limited GPU server access. We tried several spatial and temporal sampling approaches, but limited GPU availability prevented us from trying other models or additional performance tuning.

This study uses a TSN model with a ResNet-50 backbone. Although this combination achieved strong validation and test accuracy, we did not try other architectures, such as more recent video transformers or hybrid CNN–transformer models. We also did not extensively tune the training, inference, and post-processing parameters, including the SGD settings, decision threshold, smoothing window, gap merge, minimum Action duration, and end padding. We evaluated the sliding-window stride separately using strides of 8, 16, and 32 frames. Additional tuning and evaluation of other model designs could potentially improve performance.

The data were manually annotated to create the labels used for training, validation, and testing. This could introduce small differences near the boundaries between Action and No Action segments, which could affect both model training and performance evaluation. Given the imbalance between the Action and No Action class in our annotated videos we used an adaptive stride-based method for clip sampling. Future work could evaluate uniform clip-sampling intervals for both classes to determine whether different sampling densities influence model performance.

A major limitation of the study is that the research was limited to table tennis. The goal was to generate highlights from amateur table tennis videos, so the authors did not incorporate other sports. The model was not trained or evaluated in sports like tennis, badminton, or pickleball. Future work can evaluate whether this approach applies to other racket sports.

Conclusions

This study shows that machine learning can be used to generate highlights automatically from amateur table tennis videos. The TSN and ResNet-50 combination successfully identifies useful gameplay without explicitly performing ball tracking or pose estimation. It uses binary classification to detect Action and No Action segments. The primary 64-frame model achieved 88.70% validation accuracy with a Macro F1 score of 0.8865 and 92.28% test accuracy with a Macro F1 score of 0.9221. The temporal sensitivity study showed that the 16/int4 model achieved 90.10% validation accuracy with a Macro F1 score of 0.8997. These results demonstrate the effectiveness of the proposed approach for identifying active gameplay in the videos evaluated in this study.

In addition to classification performance, the system reduced the amount of video that needed to be viewed by approximately 60–70%. The generated highlight files were also considerably smaller than the original recordings, although the file-size reduction resulted from both removing No Action portions and re-encoding the highlights. This can help players and coaches review matches more efficiently without having to look through lengthy recordings.

The temporal sensitivity study showed that one could maintain similar performance while reducing computational requirements. Future work could evaluate the approach using larger and more diverse datasets, other model architectures, mobile or real-time deployment, and other racket sports. Overall, this study demonstrates a method for turning amateur table tennis recordings into shorter and more useful highlight videos for match review.

References

- 1 K. M. Kulkarni, R. S. Jamadagni, J. A. Paul, S. Shenoy. Table tennis stroke detection and recognition using ball trajectory data. arXiv preprint. arXiv:2302.09657, 2023, <https://doi.org/10.48550/arXiv.2302.09657>.
- 2 S. S. Tabrizi, S. Pashazadeh, V. Javani. Comparative study of table tennis forehand strokes classification using deep learning and SVM. IEEE Sensors Journal. Vol. 20, No. 22, pg. 13552–13561, 2020, <https://doi.org/10.1109/JSEN.2020.3005443>.
- 3 W. Guo, Z. Pan, Z. Xi, A. Tuerxun, J. Feng, J. Zhou. Sports analysis and VR viewing system based on player tracking and pose estimation with multimodal and multiview sensors. arXiv preprint. arXiv:2405.01112, 2024, <https://doi.org/10.48550/arXiv.2405.01112>.
- 4 R. Voeikov, N. Falaleev, R. Baikulov. TTNNet: Real-time temporal and spatial video analysis of table tennis. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pg. 3866–3874, 2020, <https://doi.org/10.1109/CVPRW50498.2020.00450>.
- 5 J. Bian, X. Li, T. Wang, Q. Wang, J. Huang, C. Liu, J. Cheng, J. Zhao, F. Lu, D. Dou, H. Xiong. P2ANet: A large-scale benchmark for dense action detection from table tennis match broadcasting videos. ACM Transactions on Multimedia Computing, Communications, and Applications. Vol. 20, No. 4, Article 118, 2024, <https://doi.org/10.1145/3633516>.
- 6 J. Lin, C. Gan, S. Han. TSM: Temporal shift module for efficient video understanding. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pg. 7083–7093, 2019, <https://doi.org/10.1109/ICCV.2019.00718>.
- 7 C. Feichtenhofer, H. Fan, J. Malik, K. He. SlowFast networks for video recognition. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pg. 6202–6211, 2019, <https://doi.org/10.1109/ICCV.2019.00630>.
- 8 Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, H. Hu. Video Swin Transformer. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pg. 3202–3211, 2022, <https://doi.org/10.1109/CVPR52688.2022.00320>.
- 9 L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, Y. Qiao. VideoMAE V2: Scaling video masked autoencoders with dual masking. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pg. 14549–14560, 2023, <https://doi.org/10.1109/CVPR52729.2023.01398>.
- 10 J. Hong, H. Zhang, M. Gharbi, M. Fisher, K. Fatahalian. Spotting temporally precise, fine-grained events in video. Proceedings of the European Conference on Computer Vision (ECCV). pg. 33–51, 2022, https://doi.org/10.1007/978-3-031-19833-5_3.
- 11 H. Alwassel, F. C. Heilbron, B. Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. Proceedings of the European Conference on Computer Vision (ECCV). pg. 251–266, 2018, https://doi.org/10.1007/978-3-030-01240-3_16.
- 12 T. Lin, X. Zhao, H. Su, C. Wang, M. Yang. BSN: Boundary sensitive network for temporal action proposal generation. Proceedings of the European Conference on Computer Vision (ECCV). pg. 3–19, 2018.
- 13 T. Lin, X. Liu, X. Li, E. Ding, S. Wen. BMN: Boundary-matching network for temporal action proposal generation. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pg. 3889–3898, 2019.
- 14 C. Zhang, J. Wu, Y. Li. ActionFormer: Localizing moments of actions with Transformers. Proceedings of the European Conference on Computer Vision (ECCV). 2022.
- 15 D. Shi, Y. Zhong, Q. Cao, L. Ma, J. Li, D. Tao. TriDet: Temporal action detection with relative boundary modeling. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pg. 18857–18866, 2023.
- 16 V. Valli, S. Anukarthika, S. Muhesh, B. Sri. Real-time sports action recognition using a CNN-Transformer hybrid deep learning framework. Preprints.org. 2026, <https://doi.org/10.20944/preprints202604.0150.v1>.
- 17 Z. Wang, H. Cheng, Q. Qin. The research on video analysis of key motion positions based on deep learning technology. International Journal of e-Collaboration. Vol. 21, No. 1, pg. 1–16, 2025, <https://doi.org/10.4018/IJeC.373713>.
- 18 K. M. Kulkarni, S. Shenoy. Table tennis stroke recognition using two-dimensional human pose estimation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pg. 4576–4584, 2021, <https://doi.org/10.1109/CVPRW53098.2021.00515>.
- 19 S. Yenduri, V. Chalavadi, K. M. C. Adaptive temporal aggregation for table tennis shot recognition. Neurocomputing. Vol. 584, pg. 127567, 2024, <https://doi.org/10.1016/j.neucom.2024.127567>.
- 20 K. Dong, W. Q. Yan. Player performance analysis in table tennis through human action recognition. Computers. Vol. 13, pg. 332, 2024, <https://doi.org/10.3390/computers13120332>.
- 21 H. Song, Y. Li, C. Fu, F. Xue, Q. Zhao, X. Zheng, K. Jiang, T. Liu. Using complex networks and multiple artificial intelligence algorithms for table tennis match action recognition and technical-tactical analysis. Chaos, Solitons & Fractals. Vol. 178, pg. 114343, 2024, <https://doi.org/10.1016/j.chaos.2023.114343>.

- 22 Z. K. Wei, J. R. Chang. Automatic segmentation of table tennis match video clips based on player actions for enhanced data acquisition. *Journal of Advances in Information Technology*. Vol. 15, No. 12, pg. 1374–1379, 2024, <https://doi.org/10.12720/jait.15.12.1374-1379>.
- 23 H. Faulkner, A. Dick. TenniSet: A dataset for dense fine-grained event recognition, localisation and description. 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA). pg. 1–8, 2017, <https://doi.org/10.1109/DICTA.2017.8227494>.
- 24 M. Ibh, S. Grasshof, D. Witzner, P. Madeleine. TemPose: A new skeleton-based Transformer model designed for fine-grained motion recognition in badminton. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pg. 5199–5208, 2023.
- 25 B. T. Naik, M. F. Hashmi, N. D. Bokde. A comprehensive review of computer vision in sports: Open issues, future trends and research directions. *Applied Sciences*. Vol. 12, pg. 4429, 2022, <https://doi.org/10.3390/app12094429>.
- 26 H. Xu, X. Wei, S. Wells, S. Aryal. Temporal feature distillation for label-efficient precise event spotting in sports videos. Accepted at the ACM International Conference on Multimedia (ACM MM), 2026. arXiv:2607.10998, <https://doi.org/10.48550/arXiv.2607.10998>.
- 27 Sengar SS, Mukhopadhyay S. Motion detection using block based bi-directional optical flow method. *Journal of Visual Communication and Image Representation*. 2017;49:89–103. <https://doi.org/10.1016/j.jvcir.2017.08.007>.
- 28 Lee CC, Lui PW, Gao WW, Gao Z. Detection of Rat Pain-Related Grooming Behaviors Using Multistream Recurrent Convolutional Networks on Day-Long Video Recordings. *Bioengineering*. 2024;11(12):1180. <https://doi.org/10.3390/bioengineering11121180>.
- 29 L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool. Temporal segment networks for action recognition in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 41, No. 11, pg. 2740–2755, 2019, <https://doi.org/10.1109/TPAMI.2018.2868668>.
- 30 A. Mangusheva, M. Obukhova. Training and testing an artificial intelligence model for action recognition using the MMAction2 toolkit. *E3S Web of Conferences*. Vol. 583, pg. 06016, 2024, <https://doi.org/10.1051/e3sconf/202458306016>.
- 31 K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pg. 770–778, 2016, <https://doi.org/10.1109/CVPR.2016.90>.
- 32 MultiScale Crop documentation - <https://mmapion2.readthedocs.io/en/latest/api.html>.