Action Recognition in Tennis

Diana Santos
fc64478@alunos.fc.ul.pt

Leonor Fandinga
fc64481@alunos.fc.ul.pt

Sofia Rocha
fc65111@alunos.fc.ul.pt

Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016, Lisboa, Portugal
Github: Project_Github

Abstract

This work investigates the application of Spatial Temporal Graph Convolutional Networks (ST-GCN) for action recognition in tennis, focusing on the dynamics of body joints over time. We conduct a comparative study between two training scenarios: (i) full network training, where all parameters of a pre-trained ST-GCN model are fine-tuned, and (ii) single-layer fine-tuning, where only the final classification layer is optimized while the remaining parameters are kept frozen. The performance of both models is evaluated by visualizing the loss curves, reporting top-1 and top-5 accuracy, and analysing the confusion matrix to assess class-wise prediction quality. Our results highlight the potential of pre-trained architectures for recognizing tennis movements.

1. Introduction

Understanding and identifying human movements from visual data is a central goal in the field of action recognition. This task involves assigning labels to human actions observed in video sequences or images, based on patterns of body movement, posture and timing. The system must distinguish between different types of actions — such as hitting, running or jumping — by learning how they unfold in time and space.

One of the persistent difficulties in this domain is the scarcity of large-scale, high-quality datasets. While deep learning models typically benefit from extensive data during training, many commonly used action recognition datasets remain relatively small, often comprising only a few tens of thousands of samples. This can degrade performance when models are applied to different scenarios or more complex temporal tasks.

1.1. Bibliographic review

Recent advancements in skeleton-based action recognition have centered around the use of Spatial Temporal Graph Convolutional Networks (ST-GCN), a class of models that captures both spatial relationships between joints and temporal dynamics across video frames.

The foundational work by Yan et al. [6] introduced the ST-GCN model, which represents human motion as a spatio-temporal graph where nodes correspond to body joints and edges capture both anatomical connections and temporal transitions. Unlike earlier approaches that relied on handcrafted rules, ST-GCN learns spatial and temporal patterns directly from data. The model demonstrated significant improvements over traditional methods when evaluated on large-scale datasets such as NTU RGB+D and Kinetics.

Building upon this architecture, Shi et al. [4] proposed the Two-Stream Adaptive Graph Convolutional Network (2s-AGCN) for skeleton-based action recognition. Unlike prior GCN-based approaches that rely on a manually defined and fixed graph topology, 2s-AGCN introduces a data-driven mechanism where the graph structure is adaptively learned through backpropagation. This increases the model's flexibility and ability to generalize across diverse action samples. Furthermore, 2s-AGCN employs a two-stream framework to jointly model first-order information (joint positions) and second-order information (bone lengths and directions), which are highly discriminative for human motion. Extensive experiments on large-scale datasets, such as NTU-RGBD and Kinetics-Skeleton, demonstrate that this approach significantly outperforms existing methods, achieving state-of-the-art performance.

Another extension, the Pose-Guided Graph Convolutional Network (PG-GCN), was developed by Chen et al. [2]. This model integrates multiple modalities, including pose and skeleton information, through a multi-

stream architecture. A dynamic attention mechanism enables early feature fusion, yielding robust action representations. The PG-GCN achieved state-of-the-art results on the NTU RGB+D 60 and 120 benchmarks.

Wang et al. [5] introduced the Efficient Hierarchical Co-Occurrence Graph Convolutional Network (EHC-GCN), a more computationally efficient hierarchical variant of ST-GCN. It employs separable convolution layers, dual-stream input, and channel attention to reduce computational load while maintaining accuracy. On the NTU RGB+D benchmarks, EHC-GCN improved recognition accuracy by up to 7.0% and significantly reduced FLOPs, making it suitable for resource-constrained environments.

Finally, Alsawadi and Rio [1] proposed a Skeleton-Split framework that explores different strategies for partitioning the skeleton structure to enhance ST-GCN performance. The authors compared four partitioning methods—spatial configuration, total distance, connection-based, and index-based—and found that the index-based approach achieved the highest top-1 accuracy (73.25%) on the HMDB-51 and UCF-101 datasets.

1.2. Tennis

Tennis is a sport played both individually and in pairs, with millions of athletes and fans worldwide. Generally, it consists of a confrontation between two players (or two teams of two players) separated by a net, who use a racket to hit a ball, alternating shots into the opponent's court. The objective is to keep the ball within the court boundaries and prevent it from bouncing more than once on the ground before being returned. Whenever a player fails to return the ball within the rules, the point is awarded to the opponent. In a competitive context, the variety and precision of technical movements are key elements for a player's performance. The use of different shots — such as the service, forehand, backhand, slice, or topspin — allows not only for diversifying the playing style but also for adapting to different opponents, surfaces, and conditions. Furthermore, the correct execution of movements is essential to maximize effectiveness on the court and minimize the risk of injuries. Thus, technical mastery is one of the main factors for success in modern tennis. The specific tennis movements (classes) analyzed in this study are summarized in Table 1.

Table 1. Tennis movements (classes) included in the study

Movement	Description
`backhand_slice`	Backhand shot with slice effect, using a downward movement to generate slice.
`forehand_volley`	Forehand volley (right side), executed near the net, without the ball bouncing.
`backhand_volley`	Backhand volley (left side), executed near the net, without the ball bouncing.
`kick_service`	Service with a lot of topspin, causing the ball to rise significantly after the bounce.
`slice_service`	Service with side spin (slice), which changes the trajectory of the ball after the bounce.
`smash`	Strong overhead shot, similar to a cut, used to respond to lobs.
`backhand`	Shot from the left side, with one or two hands, used when the ball comes from the opposite side to the dominant hand.
`flat_service`	Flat service, fast and with little or no spin, direct and powerful.
`forehand_openstands`	Forehand executed with an open stance (feet facing the net), common in defensive situations or fast balls.
`forehand_flat`	Flat forehand shot, with a straight trajectory and little rotation, used to speed up the game.
`backhand2hands`	Backhand shot executed with two hands, very common in modern tennis.
`forehand_slice`	Forehand shot with slice effect, less common, used for variation or control.

2. Method

The goal of the model is to classify movement sequences based on human pose data, using a convolutional network over spatio-temporal graphs (ST-GCN). Each movement sequence is represented by a temporal graph $G = (V, E)$, where:

- V represents the nodes, which correspond to 3D keypoints (joint points like head, hands, knees, etc.);
- E represents the spatial edges (natural connections between joints) and temporal edges (connections of the same nodes between consecutive frames).

Each input sample is represented by a tensor:

$$X \in \mathbb{R}^{C \times T \times V \times M}$$

Where:

- C is the number of channels (in this case 3: x, y, z coordinates);
- T is the number of frames, corresponding to 120;
- V is the number of joints (33 joints are used);
- M is the maximum number of people per clip (in this case, 1).

The spatial convolution is defined as:

$$f_{\text{out}}(v_i) = \sum_{v_j \in \mathcal{B}(v_i)} \frac{1}{Z_{ij}} W(l(v_i, v_j)) f_{\text{in}}(v_j)$$

Where:

- v_i represents the target node (a joint);
- $\mathcal{B}(v_i)$ is the set of neighboring nodes of v_i;
- f_{in} and f_{out} are the input and output features of the graph;
- W represents the learned weights associated with different types of connections, labeled by $l(v_i, v_j)$;
- Z_{ij} is a normalization factor (e.g., the number of neighbors), to ensure stability in the calculations.

This formula was taken from the article by Yan et al. [6].

The Figure 1 presents the overall architecture of the proposed approach for action recognition in tennis videos. The process begins with the extraction of video frames, from which 3D skeleton keypoints are obtained for each person. These skeleton sequences are structured into tensors and serve as input to the ST-GCN model. Within the model, spatial and temporal relationships are captured through graph construction and convolutional operations. The final output is the predicted action class, enabling precise classification of tennis movements.

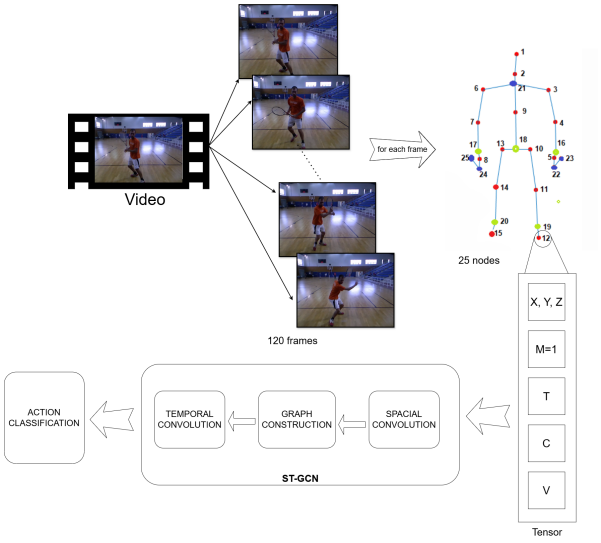

Figure 1. Confusion matrix obtained with the ST-GCN model after training with all layers of the network.

2.1. Data and Input Shape

The data used is derived from the THETIS VIDEO_RGB dataset [3]. The preprocessing steps include:

- Skeleton extraction from videos (/data/npy_videos/). Each video is converted into a .npy array containing 3D joint coordinates for each frame.
- Resampling to a fixed number of frames (/data/120_frames/). All samples are padded or truncated to exactly 120 frames ($\sim$ 4 seconds) to ensure a fixed input shape.
- A separate folder is created for each player, allowing for a later split where each player appears exclusively in only one of the sets — training, validation, or testing.
- Train/Val/Test split stored inside /data/splits/. The data is split into 70% training, 15% validation, and 15% testing, using player-based partitioning to avoid data leakage.

The data to be introduced into the model must be in a format where the set of frames is represented as .npy arrays containing the coordinates of the keypoints of the human body (obtained through pose estimation using the Python library *Mediapipe*). The typical expected structure is: (N, C, T, V, M) Where: N – Number of samples, C – Number of channels, T – Number of frames, V – Number of vertices, M – Number of people.

2.2. Implementation Details

The implementation is carried out with the *MMAction2* framework, using a configuration adapted to work with 3D data (three-dimensional keypoints) for action recognition. The specified model is of type RecognizerGCN, utilizing

ST-GCN as the backbone with the nturgb+d layout and three input channels corresponding to the spatial coordinates x, y, and z. The classification head (GCNHead) operates over 256 channels and is configured to recognize 12 distinct classes, employing the CrossEntropyLoss function for optimization.

During data pre-processing, the pipeline applies several steps to prepare the input for the model. Initially, the PreNormalize3D operation standardizes the 3D skeleton data. Next, skeletal features are generated using GenSkeFeat with the *'j'* (joint) feature from the nturgb+d dataset. UniformSampleFrames is then used to sample 100 frames per video, ensuring temporal consistency. The PoseDecode step transforms the keypoint data into a format interpretable by the model. The data is subsequently formatted for GCN input, considering up to one person per sequence, and finally packaged for action recognition tasks. These preprocessing steps are consistently applied across the training, validation, and testing pipelines, with minor parameter adjustments for evaluation modes.

Data loading is managed by PoseDataset objects, each referencing specific annotation files for training, validation, and testing. The batch size is set to 64, and data loading is parallelized with two workers and persistent worker processes enabled for efficiency.

Model training utilizes the SGD optimizer with a learning rate of 0.1, momentum of 0.9, weight decay of 0.0005, and Nesterov acceleration. The learning rate is scheduled using CosineAnnealingLR, with T_max set to 16 and eta_min to 0, providing a smooth and gradual decrease in the learning rate throughout training. The training process follows an EpochBasedTrainLoop with a maximum of 50 epochs, with validation starting at the first epoch and occurring every epoch thereafter.

Model performance is evaluated using top-k accuracy metrics (top-1 and top-5), as well as mean class accuracy. Pre-trained weights from OpenMMLab, trained on NTU RGB+D with 3D keypoints, are used for initialization. All experiment artifacts, including checkpoints and logs, are saved in the designated working directory.

3. Results

In the performance analysis, two training strategies applied to the model were compared, namely complete network training and training with only the last layer. Comparing these approaches is essential to understand how flexibility in updating parameters influences the model's ability to learn and converge during the optimization process.

3.1. Confusion Matrix

Figure 2 represents the confusion matrix obtained after training the model with all layers of the network. This allows us to evaluate the classification made by the model for the different tennis actions mentioned in Table 1.

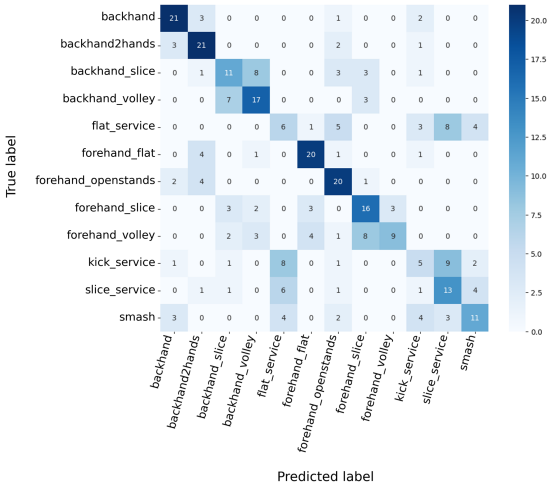

Figure 2. Confusion matrix obtained with the ST-GCN model after training with all layers of the network.

The model demonstrates good performance in identifying some classes, such as forehand_flat, forehand_openstands, and backhand, with most of their samples correctly classified (high values along the diagonal). These actions likely present distinctive patterns that the model can easily capture.

However, the model struggles with classes that share visual or temporal similarities. For instance, backhand_slice and backhand_volley are often confused, suggesting that the network finds it challenging to distinguish subtle variations within backhand motions. A similar pattern is observed among the service types — particularly flat_service, kick_service, and slice_service — where misclassification are higher probably due to similarities in motion and posture.

The smash class shows reduced accuracy, often being confused with kick_service and flat_service. This may be attributed to a smaller number of training examples for this action or greater variability in its execution.

Overall, while the model performs well on more distinct classes, the confusion matrix reveals its limitations when it comes to fine-grained discrimination of similar actions, particularly among serves and backhand variations.

Figure 2 shows the confusion matrix obtained after training the ST-GCN model with only the last layer, keeping the others frozen. The diagonal elements of the matrix represent correct predictions, with darker shades indicating greater accuracy. Notably, actions such as forehand_volley, kick_service and forehand_slice achieved good rankings, with 11, 10 and 10 correct predictions respectively. These results suggest that the representations learned by the pre-

trained model retain sufficient discriminative information for certain types of actions, even when the last layer is used.

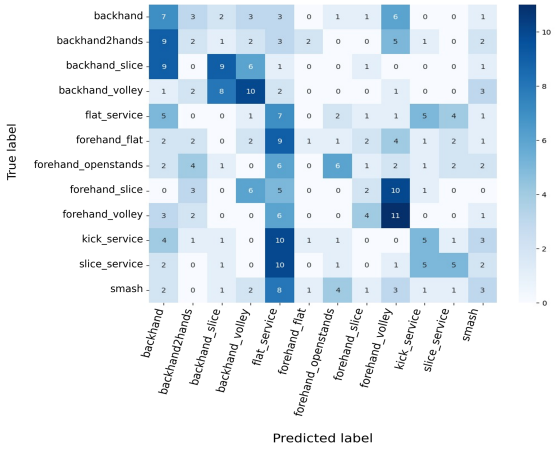

Figure 3. Confusion matrix obtained with the ST-GCN model after single-layer fine-tuning.

On the other hand, the matrix reveals notable confusion in several categories. For example, backhand_slice and backhand are often confused. Similarly, flat_service and slice_service show overlapping predictions, reflecting the challenges in distinguishing similar service actions.

Thus, although the model trained with the last layer presents some correct classifications, the model trained with all layers has a much more reliable and accurate performance.

3.2. Top-1 and Top-5 accuracy

Figure 4 compares the Top-1 and Top-5 accuracy achieved by two training strategies: full network training and single-layer fine-tuning.

The blue line represents the Top-1 accuracy during full network training, while the dashed blue line shows the corresponding Top-5 accuracy. Both metrics increase rapidly and steadily throughout training, reaching values close to 100%, which indicates that the model learns to correctly classify most samples with high confidence.

In contrast, the red lines correspond to single-layer fine-tuning. The solid red line (Top-1 accuracy) remains low and unstable, rarely exceeding 40%. The dashed red line (Top-5 accuracy) performs better but still lags behind the full training scenario, stabilizing around 80%.

These results demonstrate that full network training yields significantly better performance, both in terms of convergence and generalization. Limiting updates to a single layer restricts the model's ability to learn meaningful representations, leading to suboptimal accuracy.

3.3. Loss Curve

Figure 5 illustrates the loss curves of the trained model obtained to compare training using a complete network and only the last layer of that same network. The horizontal axis represents the training steps, while the vertical axis shows the loss value.

The blue curve corresponds to the full network training, where all layers of the model are updated during optimization. In this setting, the loss decreases rapidly and consistently, showing a stable convergence pattern. The final loss approaches zero, indicating that the model effectively learns from the data and minimizes the training error.

In contrast, the red curve represents the single-layer fine-tuning approach, in which only one layer is updated while the remaining layers are frozen. This configuration leads to a significantly slower and more limited decrease in training loss. The loss plateaus early, remaining above 1.6 throughout the training process. This suggests that restricting training to a single layer hampers the model's capacity to adapt and fit the training data properly.

These results demonstrate that full network training provides substantially better convergence and learning capacity, while single-layer fine-tuning is insufficient for capturing the complexity of the task.

4. Conclusion

This study explored the application of Spatial Temporal Graph Convolutional Networks (ST-GCN) to recognize specific actions from tennis videos. The results demonstrate that while using a pre-trained framework with only the final layer trained provides moderate performance for some well-defined actions, it falls short in classifying more complex or visually similar movements. In contrast, training the full network produces significantly better classification accuracy, faster convergence, and lower training loss, as shown in the confusion matrices, top-k accuracy metrics, and loss curves.

These findings highlight the importance of allowing the model to adapt its internal representations when transferring to a new domain-specific task, such as tennis action recognition.

Despite efforts to build a custom tennis stock dataset, it was not possible to complete this process due to time constraints, but it can serve as a basis for future work.

References

[1] Motasem Alsawadi and Miguel Rio. Skeleton-split framework using spatial temporal graph convolutional networks for action recognition. In *International Conference on Advances in Computing and Data Sciences*, 2022. 2

[2] Han Chen, Yifan Jiang, and Hanseok Ko. Pose-guided graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:2101.05270*, 2021. 1

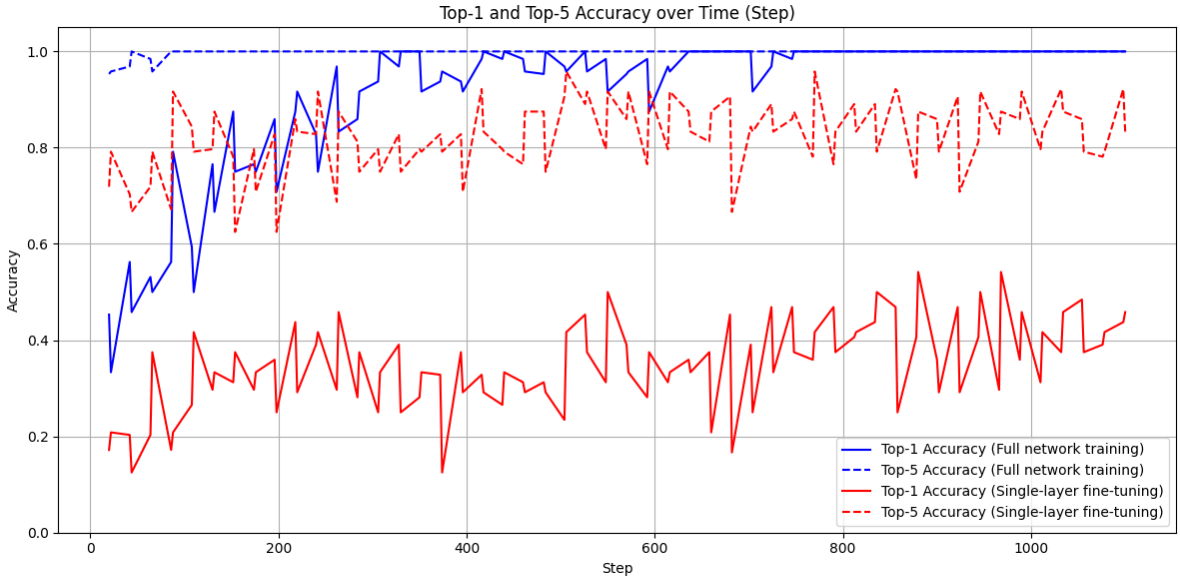

Figure 4. Evolution of top-1 and top-5 accuracy during training in both approaches.

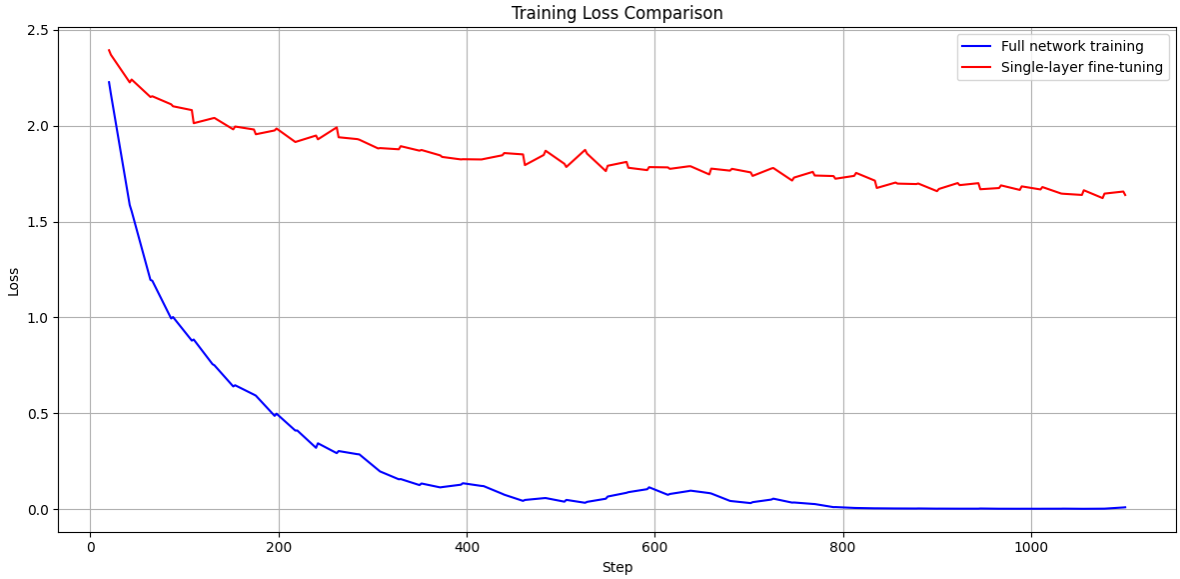

Figure 5. Comparison of the loss during training between full network training and fine-tuning of a single layer.

[3] S. Gourgari, G. Goudelis, K. Karpouzis, and S. Kollias. Thetis: Three-dimensional tennis shots–a human action dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 676–681. IEEE, 2013. 3

[4] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019. 1

[5] H. Wang, Y. Bai, J. Xu, D. Yang, and L. Xu. Ehc-gcn: Efficient hierarchical co-occurrence graph convolution network for skeleton-based action recognition. *MDPI Sensors*, 2022. 2

[6] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 1, 3