Allez Go: Computer Vision and Audio Analysis for AI Fencing Referees

ABSTRACT

The gradual increase in online fencing videos over the past decade has allowed for novel technical projects in fencing that rely heavily on data, such as artificial intelligence. This study resulted in a state-of-the-art lightweight Temporal Convolutional Network to referee fencing bouts and classify actions as either a touch for the fencer on the left or the fencer on the right. To address this problem, we developed a pose estimation and audio analysis approach to autonomously referee fencing bouts. Using a custom dataset of international level fencing from the last 7 years, including ~4000 unique clips, our model achieved an accuracy of 89.1%, a 20% increase over previous state-of-the-art models. This model leverages advancements in human pose estimation to extract the position of both fencers and avoids high computational loads typically associated with CNNs. Additionally, it uses a novel technique to solve the issue of blade contact, a key component of refereeing fencing that was generally unaddressed in previous works. Our novel solution uses audio to 'listen' for the sound of blade contact rather than attempting to identify it visually.

1. Introduction

1.1 Relevant Rules of Fencing

Fencing is generally a simple sport where the fencer that lands a valid hit with their blade onto their opponent scores a point. However, complexities arise when both fencers hit each other at the same time. In this situation, the fencing rulebook states that a referee will determine who had right of way during the action and will award the point to that fencer. The concept of right of way is difficult to explicitly define, although it is generally given to whichever fencer starts their attack firsts, and transfers to the other fencer if their attack fails. The beginning of an attack is identified by a combination of various factors such as the fencer's direction of movement, whether or not their sword arm is extending, their body language, etc. These factors are impossible to quantify and referees may vary in how they interpret these movements and priority.

An important consideration while refereeing fencing is blade contact. Right of way, also known as priority, transfers from one fencer to another when their attack fails. One of the most common reasons an attack fails is due to a fencer successfully blocking and deflecting an incoming attack using their blade. This is known as blade contact and generally, blade contact is synonymous with the transfer of priority, making it an important cue for referees to look out for.

1.2 Importance

Technology in fencing is generally an underdeveloped field and automated referees present potentially significant benefits to the sport. Automated referees will offer a more consistent call compared to a group of human referees with slightly different interpretations of the fencing rules. Since the overall fencing referee community is also relatively small, AI referees may be useful for the training of new referees where veteran referees are unavailable.

An automated refereeing system may also be important in standardizing the calls of human referees. An AI system could flag calls from human referees that it disagrees with, which could later be reviewed by a panel of veteran referees. If the panel agrees with the human referees, that data would be used to improve the AI referee, while if the panel disagrees with the human referee, they could offer corrective feedback to that referee.

Another promising application lies in sports broadcasting. Priority is generally not understood by spectators and an AI referee can display visual cues that indicate which fencer has priority so that spectators can understand how priority works and how it influences a fencer's decisions during a bout.

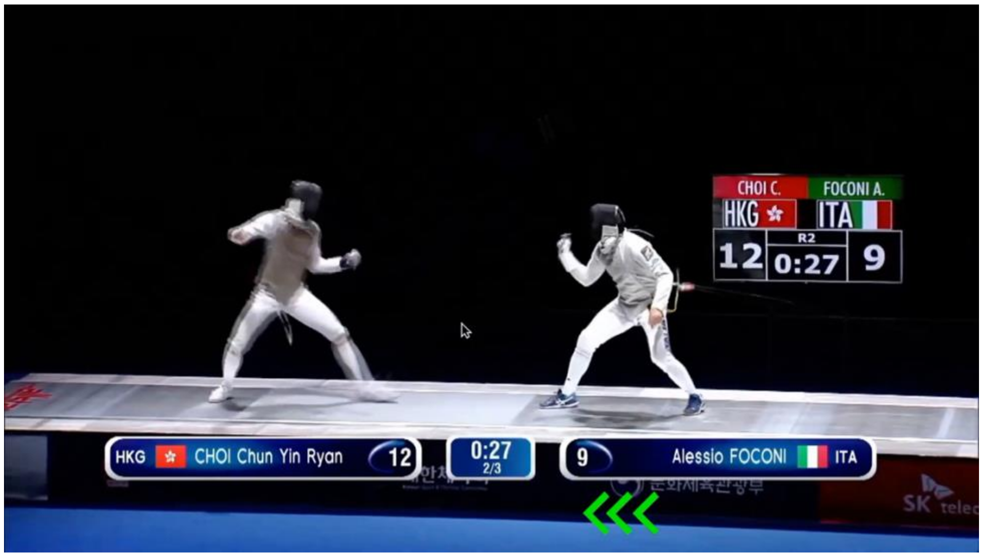


Figure 1: AI-generated arrows show which fencer has priority.

1.3 Temporal Convolutional Networks

This paper uses temporal convolutional networks (TCNs) for temporal analysis rather than a standard LSTM or GRU approach. Compared to other temporal architectures, TCNs feature a flexible receptive field that can exhibit longer memory while also maintaining faster execution times. The size of a TCN's receptive field can be calculated using equation 1, where K_{size} is the kernel size, N_{stack} is the number of stacks, n is the number of layers and d_i is the dilation value at the i*th* layer [2].

$$R_{field} = 1 + 2 \cdot (K_{size} - 1) \cdot N_{stack} \cdot \sum_{i}^{n-1} d_i$$

Equation 1: Receptive field formula

TCNs calculate convolutions in parallel rather than sequentially, resulting in both a faster processing time and lower memory usage [3]. Additionally, TCNs have shown better performance on various sequence-based tasks than traditional RNNs [4].

2. Previous AI Work in Fencing

GitHub user Sholtodouglas used an LSTM-based approach with optical flow to detect motion and achieved results of 60% accuracy [5]. His model leverages transfer learning using Inception V3 with a custom recurrent network at the top. Sholtodouglas also pioneered a new method of data collection specifically for fencing which inspired our data collection process.

A more recent approach from Alexandre Pageaud uses OpenPose, a pose estimation program for preprocessing, which is later fed into a CNN-LSTM model that achieved 70% accuracy [6]. These results are similar to early work in this study, which used pose estimation and a TCN model. Pageaud cites the lack of information on blade contact as a major shortcoming, which is addressed in this paper using a novel audio approach.

Since 2012, Fencing Tracking and Visualization System has been developing methods to track blades for sports broadcasting and spectator displays, with two notable solutions. The first, published in 2018, uses IR reflectors taped onto a fencer's blade and IR cameras to track the position of blades [7]. The second, developed in 2020 uses a purely camera-based approach, with no additional equipment attached to the fencers. Instead, Yolo V3 is used for object detection with footage from a large array of cameras filtered using Lidar [8]. Both of these methods produced high-accuracy blade tracking and have been used in some high-level fencing tournaments.

3. Dataset

A custom dataset was created for this paper using online videos of 33 international competitions dating back to 2015. These videos were all recorded from cameras located near the center of the fencing piste on a relatively tall tripod.

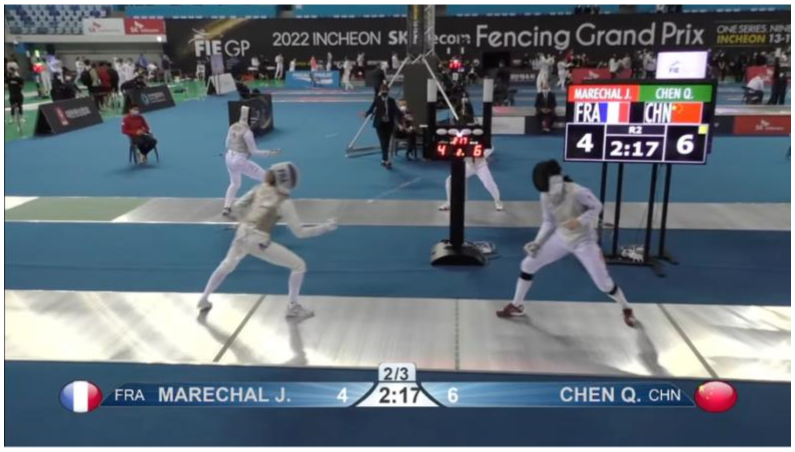


Figure 2: Example frame from the dataset.

Another important similarity among the videos is that they feature the same scoreboard. When fencers score a hit, the scoreboard will show a corresponding light. A valid hit from the fencer on the left is signified by a red light, a valid hit from the fencer on the right is represented by a green light, and invalid hits from either fencer are shown with white lights.

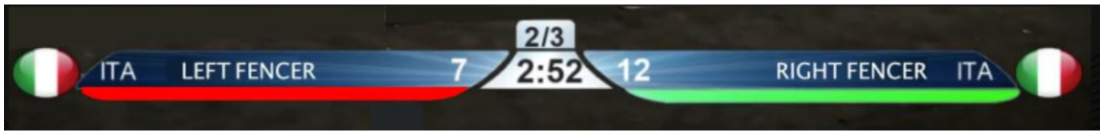


Figure 3: Scoreboard lights when both fencers score a valid hit.

Since refereeing is only applicable when both fencers score a hit, a program was used to go through each video, detect when both lights go off, and then save the 2 seconds before that hit. Labeling each clip was also done autonomously using the scoreboard and a digit recognition program. There are three scenarios that were labeled. Firstly, if both fencers scored a valid hit, priority was given to whichever fencer saw a score increase. In figure 3, if the score changed to 8-12, then priority was given to the left fencer. The second scenario is if the left fencer scores an invalid hit while the right fencer scores a valid one. In this case, if the score does not change, that means priority was given to the fencer with an invalid hit and if it does change, priority was given to the fencer with a valid hit [5]. The third scenario is the reverse of scenario two, where the right fencer scores an invalid hit, and the left fencer scores a valid hit. This algorithmic approach allowed us to cut and label hundreds of hours of fencing footage with very few man-hours.

The collected dataset featured 4500, two-second long clips that were labeled as either "Left" or "Right". Each clip also had its corresponding audio saved. Data augmentation was also performed by flipping each clip horizontally and reversing its label to double the final dataset to 9000 clips. Although the dataset was relatively balanced before data augmentation (49% Left / 51% Right), horizontal data flipping resulted in a perfectly balanced dataset. 80% of the dataset was used for training while 20% was set aside for validation testing.

4. Methods

Both video and audio data had features extracted using off-the-shelf packages to save time during model training and ensure only relevant features were extracted.

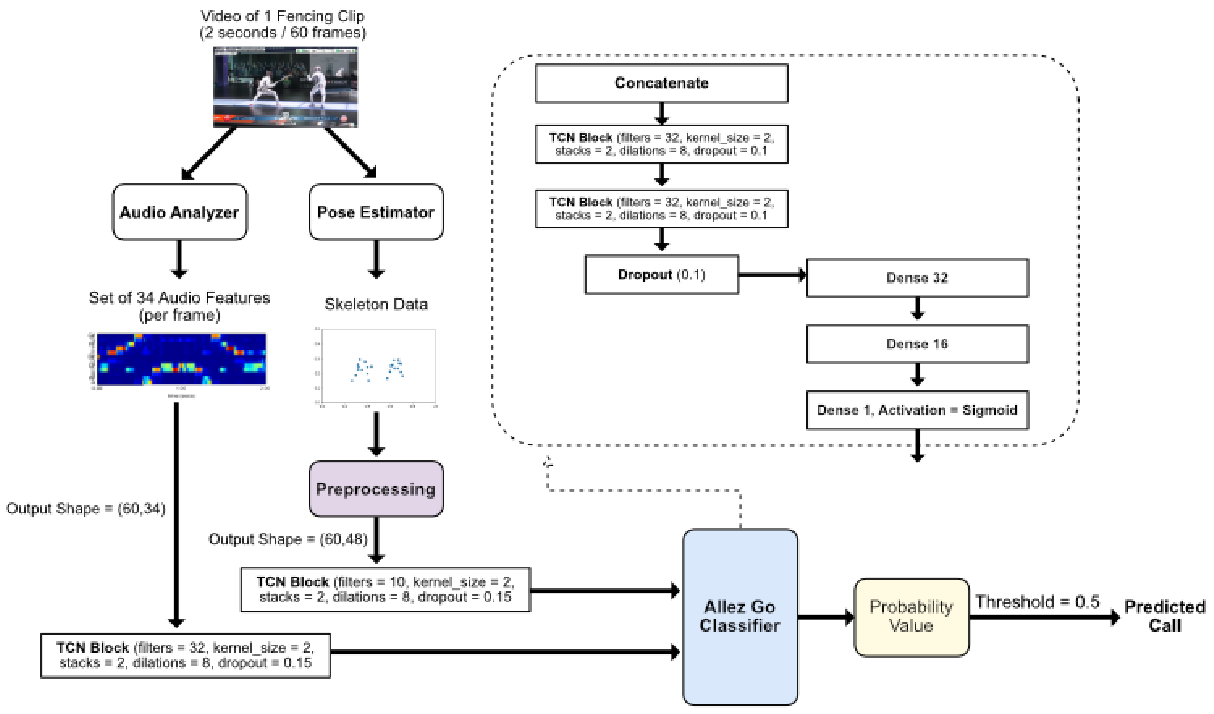

Video of 1 Fencing Clip
(2 seconds / 60 frames)
Audio Analyzer
Pose Estimator
Set of 34 Audio Features
(per frame)
Skeleton Data
Preprocessing
Output Shape = (60,34)
Output Shape = (60,48)
TCN Block (filters = 10, kernel_size = 2, stacks = 2, dilations = 8, dropout = 0.15
TCN Block (filters = 32, kernel_size = 2, stacks = 2, dilations = 8, dropout = 0.15
Concatenate
TCN Block (filters = 32, kernel_size = 2, stacks = 2, dilations = 8, dropout = 0.1
TCN Block (filters = 32, kernel_size = 2, stacks = 2, dilations = 8, dropout = 0.1
Dropout (0.1)
Dense 32
Dense 16
Dense 1, Activation = Sigmoid
Allez Go Classifier
Probability Value
Threshold = 0.5
Predicted Call

Figure 4: Overall Allez Go Network Architecture

4.1 PoseNet

Visually, the most important feature to extract was the position and stance of both fencers. A traditional CNN approach was tested but proved to be too computationally intensive and required unworkably long training times. Instead, a human-pose-estimator, PoseNet was used for visual feature extraction.

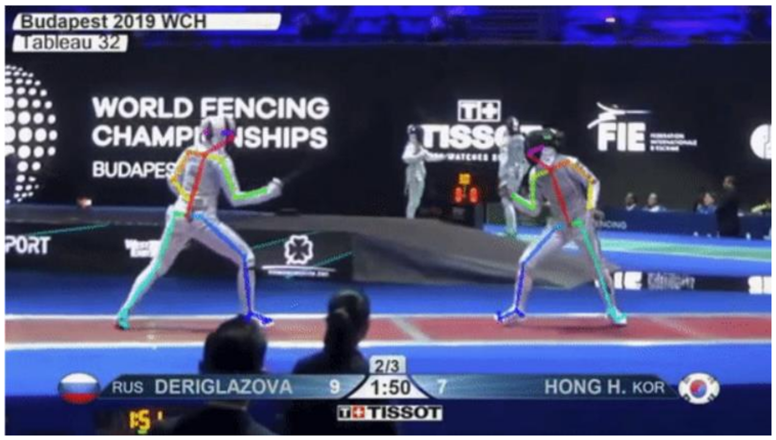


Figure 5: Example skeleton pose outputted by PoseNet

PoseNet is a real-time pose-estimating algorithm developed by TensorFlow. The program outputs the pose of all human figures within a frame with a corresponding confidence value [9]. However, to retain only the pose of both fencers, only the two poses with the highest corresponding confidence value were kept. This approach generally worked although a major issue arose when referees cross the camera's field of view. PoseNet generally detects the referee, who is closer to the camera, with higher confidence than one of the fencers. These scenarios were manually filtered out from the dataset. Additionally, unnecessary information, such as the position of the fencer's toes or head were removed to combat overfitting.

Table 1: Clip counts for each class before and after manual filtering

	Before Filtering	**After Filtering**	**Percent Filtered**
Left	3829	2441	36.2%
Right	4066	2542	37.5%
Total (before data augmentation)	7895	4983	36.9%

Unfortunately, PoseNet does not come in with a built-in pose tracker. This means that the fencer on the left will not always be given the same index in a list. A rudimentary tracking system was implemented where the pose with the smaller median of x-coordinates would be placed first in the list, resulting in the dataset having a consistent format of [left fencer, right fencer]. Additionally, all values were normalized to fit between 0 and 1.

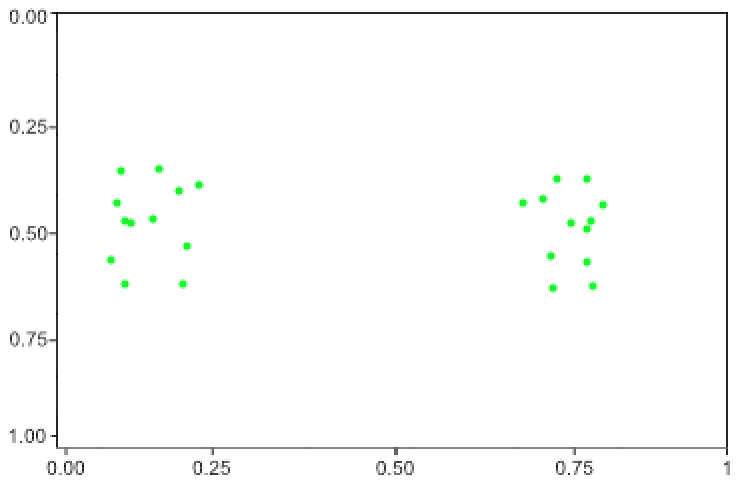


Figure 6: Plot of preprocessed X-Y coordinates of fencer's poses

4.2 PyAudio Analysis

Short-term audio features were extracted using the PyAudio Analysis library. This library extracts 34 different audio features such as zero crossing rate, MFCCs, and spectral flux [10]. Features were extracted in 0.33-second time intervals to match the frame rate of the video data. This encourages the model to analyze the visual and audio data in tandem. One benefit of this is to distinguish between audio from the bout in question and background audio from other bouts. If the visual position of both fencers suggests that blade contact may have occurred and in the audio, there is the sound of blade contact then it is highly likely that blade contact occurred. On the other hand, if visually, the fencers are far away and unlikely to beat blades the sound of blade contact is likely to be from another bout in the background.

Table 2: Features extracted by PyAudioAnalysis [10]

Feature ID	Feature Name	Description
1	Zero Crossing Rate	The rate of sign changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values (normalized)
3	Entropy of Energy	Can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
9-21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are distributed according to the mel-scale.
22-33	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western music (semitone spacing).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

A computer-vision approach to detecting blade contact was not chosen due to the relatively low image quality of our dataset, making it difficult to see blades. Additionally, pursuing a computer-vision approach to detecting blade contact would require a time-intensive, manual, frame-by-frame labeling of videos. While tracking blade’s visually is technically possible, as proven by Fencing Tracking and Visualization System’s work, their methods are unfeasible for training a referee since both are relatively novel and have not seen widespread use, leading to insufficient data to train an AI algorithm with.

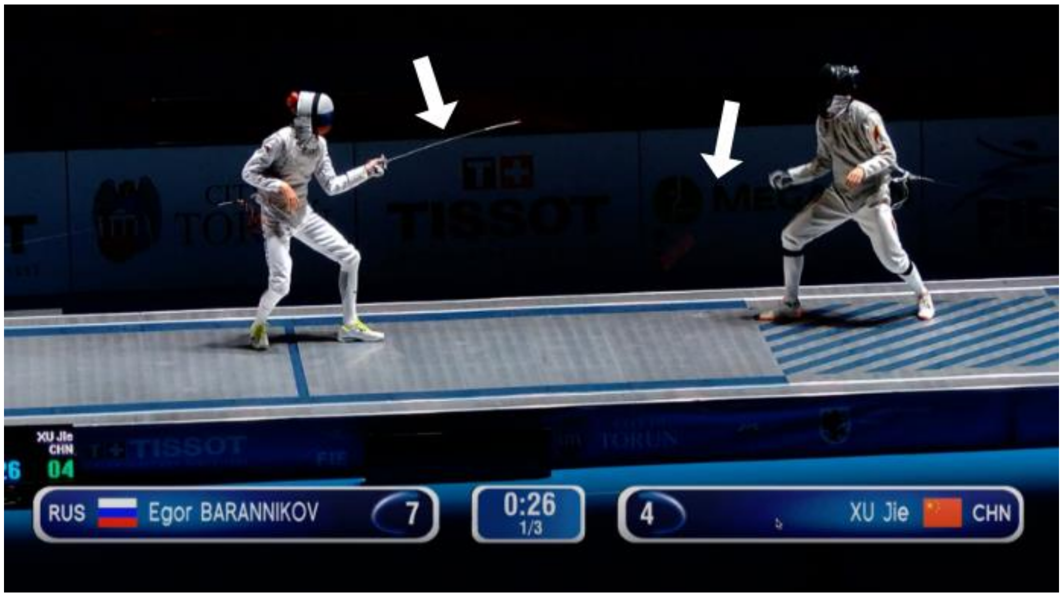


Figure 7: Example of easy-to-see blades (on the left) and difficult-to-see blades (on the right). It should be noted that this example frame has relatively good lighting compared to most of the dataset.

4.3. Model Architecture

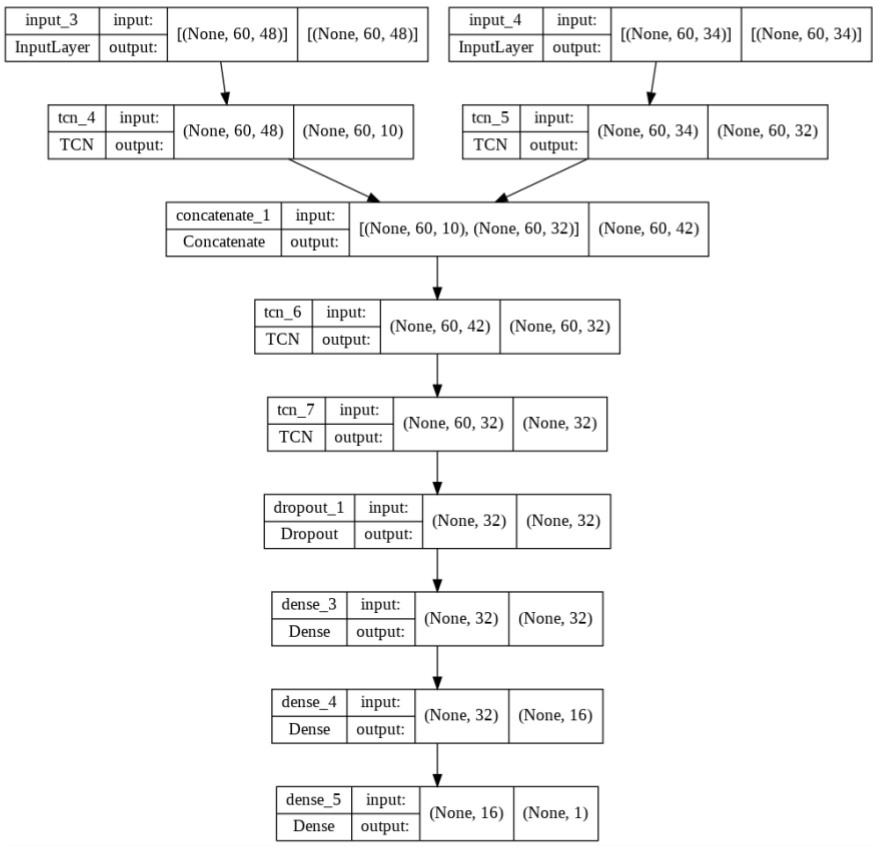
An initial model that only had visual input produced results of around 65% accuracy, which increased to 68% after intensive hyperparameter tuning, adjusting dropout intensity, number of layers, and other parameters. A confusion matrix of the visual-only model shows that it struggles on 'priority left' clips.

Table 3: Confusion matrix of visual-only model

		Predicted Label	
		Right	**Left**
True Label	**Right**	720	185
	Left	418	491

After manual analysis, it was concluded that the presence of blade contact was a common feature among clips that the model incorrectly predicted, matching conclusions by Pagaeud in his work. Both the confusion matrix and manual analysis support the theory that the model is missing critical information to distinguish between certain 'Left' or 'Right' calls. The visual-only model defaults to labeling these undistinguishable clips as 'right' (label = 0), resulting in an unbalanced confusion matrix. It was concluded that blade contact was the missing information and that audio should be added to the architecture.

The final model architecture is shown in figure 8, with two sub-models, one for video and one for audio concatenating into a combined model. The first four layers feature temporal convolutional networks while the final classification is done by dense layers. Hyperparameter tuning was conducted to combat overfitting, which was a systemic issue throughout the entire development process. Nearly all model architectures tested achieved near-perfect train accuracy with varying degrees of test accuracy.


input_3 InputLayer input: output: [(None, 60, 48)] [(None, 60, 48)]
input_4 InputLayer input: output: [(None, 60, 34)] [(None, 60, 34)]
tcn_4 TCN input: output: (None, 60, 48) (None, 60, 10)
tcn_5 TCN input: output: (None, 60, 34) (None, 60, 32)
concatenate_1 Concatenate input: output: [(None, 60, 10), (None, 60, 32)] (None, 60, 42)
tcn_6 TCN input: output: (None, 60, 42) (None, 60, 32)
tcn_7 TCN input: output: (None, 60, 32) (None, 32)
dropout_1 Dropout input: output: (None, 32) (None, 32)
dense_3 Dense input: output: (None, 32) (None, 32)
dense_4 Dense input: output: (None, 32) (None, 16)
dense_5 Dense input: output: (None, 16) (None, 1)
Figure 8: Model Architecture

Experimentally, a temporal convolutional network was proven to perform better than traditional temporally aware model architectures such as LSTMs and GRUs. It is suspected that this is due to TCN's flexible receptive field, allowing it to be the exact length of our input sequence. Each clip in our dataset is 2 seconds long with 30 FPS, leading to a total sequence length of 60 steps. The receptive field of Allez Go's TCN was calculated to be 61 using equation 1 with kernel size = 2, number of bases = 2, number of layers = 4, and dilations 1,2,4,8 respectively. . This allows TCNs to avoid overfitting to a greater extent than LSTMs and GRU by allowing us to reduce the model's size while still maintaining full coverage. Thanks to this, the final architecture is relatively lightweight, with only 109,315 parameters. Additional steps taken to combat overfitting were the inclusion of several dropout layers with values ranging from 0.01 to 0.15 and L2 regularization. Due to TCN's ability to train in parallel [3], the model was trained in less than three hours on a Google Colab Tesla T4 for 5000 epochs.

Table 4: Performance comparison of similarly sized architectures

Method	Accuracy	Recall	Precision	F1-Score
LSTM	79.4%	0.769	0.810	0.789
GRU	63.9%	0.462	0.682	0.551
TCN (Visual Only)	66.8%	0.540	0.726	0.620
TCN (Allez Go)	**89.1%**	**0.879**	**0.886**	**0.882**

6. Results

The final model achieves 89% accuracy after modest hyperparameter tuning: mainly focused on dropout intensity and the number of connected layers. Some common features among clips incorrectly predicted by the model include a complete lack of audio and possible blade contact that is inaudible through the microphone. A confusion matrix shows that the model performs equally well on both “Left” and “Right” calls.

Table 5: Confusion matrix for Allez Go

		Predicted Label	
		Right	**Left**
True Label	**Right**	802	103
	Left	110	799

It should also be noted that Allez Go suffers from overfitting, with a near-perfect train-set accuracy. Despite this clear sign of overfitting, the model generalizes well to the test set.

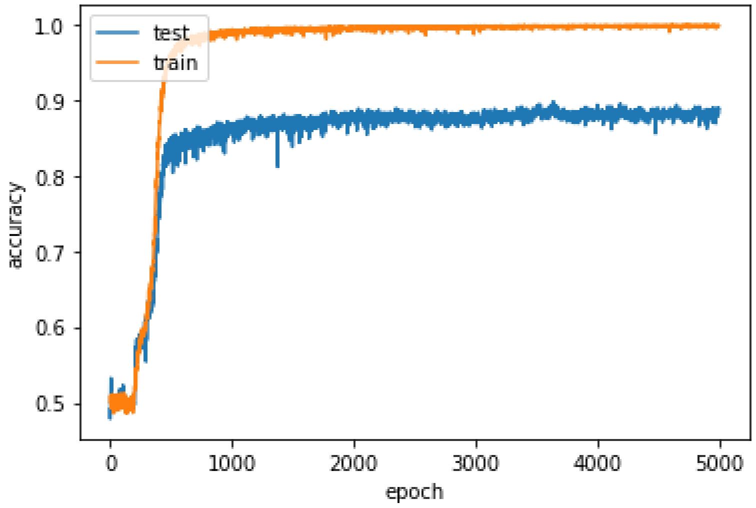


Figure 9: Train and test accuracy plot

Compared to previous state-of-the-art models, our main innovation was the inclusion of audio for the purpose of tracking blade contact. Previous work similarly used pose estimation and recurrent networks and achieved roughly ~70% accuracy, matching our visual-only model’s performance as well. However, after including audio in the architecture, our model saw an immediate jump to roughly 90% accuracy.

Table 6: Comparison with previous works

Name	Accuracy*	Approach
Sholtodouglas [5]	~60%	Recurrent, Pre-Trained Inception V3, and Optical Flow
Fencing Matches AI [6]	~70%	Pose Estimation and CNN + LSTM model
GalDude33 [11]	~70%	Pose Estimation and Optical Flow
Allez Go (Ours)	89.1%	Pose Estimation and Audio

*only approximate accuracies were provided by previous works

7. Conclusion

This paper presents two novel techniques to automate fencing refereeing that resulted in a 20% increase over previous state-of-the-art models. Our AI fencing referee highlights the potential of TCNs in certain use cases to perform better than traditional recurrent architectures. Additionally, in a field generally focused on computer vision, our model shows that sound can also be an important aspect to consider for automated sports refereeing or other scenarios where spatial awareness is required and image quality is poor.

In the future, we would like to train a model to recognize specific actions, rather than just giving a general 'Left' or 'Right' call. Further improvements could also include a more precise approach to extract only fencers from the PoseNet outputs. Gradual improvements to the model could also be made by retraining it as new fencing footage becomes available.

The work presented in this paper is available for use via an online web app. In terms of performance, the model is able to run on only a CPU and 800 MB of memory, despite working with relatively large video files. Each frame takes roughly 250 milliseconds to process, with the slowest step being the pose-estimator.

9. Acknowledgements

The author of this paper would like to thank Paul Jolly and Anna Orosz from InSpirit AI for their assistance in navigating the publication process and members of the fencing community that have shown support and excitement for our work in AI fencing referees.

References

[1] Chirashnya, I. (2020, August 18). *A dummy's guide to right of way or priority in fencing*. Academy of Fencing Masters Blog. Retrieved from https://academyoffencingmasters.com/blog/a-dummys-guide-to-right-of-way-or-priority-in-fencing/

[2] Philipperemy. (2022, July 23). *Philipperemy/Keras-TCN: Keras temporal convolutional network.* GitHub. Retrieved from https://github.com/philipperemy/keras-tcn

[3] Zhu, K., Wong, A., &amp; McPhee, J. (2022). FenceNet: Fine-grained footwork recognition in fencing. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw56347.2022.00403

[4] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271, 2018.

[5] Douglas, S. (2018, November 30). *Sholtodouglas/fencing-ai: Using deep learning to referee fencing*. GitHub. Retrieved from https://github.com/sholtodouglas/fencing-AI

[6] Pageaud, A. (2019, September 14). *Sport : Fencing matches ai*. Kaggle. Retrieved from https://www.kaggle.com/datasets/alexpgd/sport-fencing-matches-ai

[7] Takahashi, M., Yokozawa, S., Mitsumine, H., Itsuki, T., Naoe, M., & Funaki, S. (2018). Sword Tracer. *ACM SIGGRAPH 2018 Talks*. https://doi.org/10.1145/3214745.3214770

[8] Hanai, Y., McDonald, K., Horii, S., Kera, F., Tanaka, K., Ishibashi, M., & Manabe, D. (2021). Fencing Tracking and visualization system. *SIGGRAPH Asia 2021 Real-Time Live!* https://doi.org/10.1145/3478511.3491310

[9] *Real-time human pose estimation in the browser with tensorflow.js*. The TensorFlow Blog. (2018, May 7). Retrieved from https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html

[10] Tyiannak. (2022, April 19). *Tyiannak/pyaudioanalysis: Python Audio Analysis Library: Feature extraction, classification, segmentation and applications*. GitHub. Retrieved from https://github.com/tyiannak/pyAudioAnalysis

[11] Dudovitch, G. (2019, January 22). *Galdude33/fencing-ai: Using deep learning to referee fencing*. GitHub. Retrieved from https://github.com/GalDude33/fencing-AI