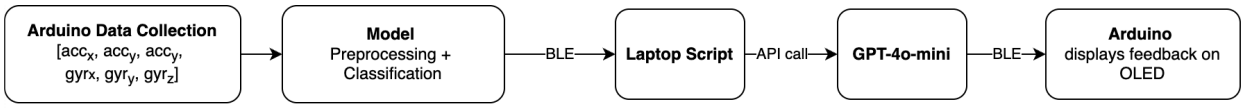
18848 Tiny Tennis Coach Final Report Ann Kang and Arti Jain

Abstract

Tiny Tennis Coach is a wearable device strapped to a player's wrist that can analyze a player's swing and provide targeted feedback to help improve a player's form. Inspired by our own experiences playing tennis, we aimed to create a convenient and real-time learning aid for beginners to get feedback on their swing. Tiny Tennis Coach analyzes accelerometer and gyroscope information from a player's swing and uses an edge-AI model deployed on an Arduino Nano 33 BLE Sense to classify the swing type and quality, then generates textual advice using a large language model (GPT-4o-mini), which is displayed on an OLED screen for the player to read. In this report, we discuss our methodology, results, and lessons learned.

High Level Block Diagram



Dataset and Data Cleaning

We aimed to classify forehand, backhand, and serve swings while also further classifying the quality of forehand and backhand swings with accelerometer and gyroscope data. We determined that the most common beginner mistakes on these swings were on the take-back and/or follow-through. Thus, we further divided the forehand and backhand classes into the subcategories, resulting in the following classes:

- Good serve
- Bad serve
- Forehand_XXXX*
- Backhand_XXXX*
- Idle

*Each 'X' is a binary indicator (0 or 1) describing whether the player had a (1) good/bad overall stroke, (2) presence/absence of takeback (3) presence/absence of follow-through, (4) good/bad take-back form, respectively.

Thus, we originally had a total of 19 classes. The criteria for a "good" take-back form included demonstrating the correct swing path on the take-back. The criteria for a "good" overall forehand or backhand included demonstrating the correct take-back form, exhibiting a follow-through on the swing, and producing a valid hit in-bounds. The criteria for a "good" serve included demonstrating the correct swing path and producing a valid hit into the service box. Our data collection technique was to use first-aid gauze to strap the Arduino with a battery to the player's wrist, maintaining consistent alignment of the Arduino's orientation across sessions and in deployment. We collected motion data from the Arduino, which is relayed to a laptop over BLE and saved to a CSV file while simultaneously filming the player. We had individuals rally back and forth,

repeatedly perform one movement, and mime the movements without a ball to collect data for different types of swings and swing qualities. To clean the data, we used the video to match up and label the movements in the motion data.

Model Training and Performance

We preprocessed raw data using spectral analysis in Edge Impulse. The accelerometer data was on a much smaller scale (1 to 3 g's) than the gyroscope data (hundreds of degrees/second), so we scaled the accelerometer data to achieve better model performance. We used FFT to extract frequency information, and found that a larger FFT helped increase the resolution of frequency detail and better captured changes in movement and vibrations in our dataset. Our classifier architecture is a standard feed-forward neural network with 3 dense layers decreasing in number of hidden units followed by one layer of dropout. We included primarily dense layers since our model takes in spectral data and does not need spatially aware layers such as convolution or reshaping. We found that 3 dense layers was the best balance between a complex enough model that could classify and rate the quality of swings, while also being simple enough to deliver results in real-time. We included dropout to prevent the model from overfitting.

Because of limited time, we did not finish cleaning all of our data and thus some of our classes had very few samples. We had to consolidate the 16 backhand and forehand classes into two overall "good" and "bad" classes, reducing the number of classes from 19 to 7. Below are the results of our training and testing accuracy with 7 classes across a total of 714 training samples and 184 test samples and an 80/20 train/test split (Note that 0000 represents "bad" and "1111" represents good").

Training Confusion Matrix

	BACKHAND_0000	BACKHAND_1111	BAD_SERVE	FOREHAND_0000	FOREHAND_1111	GOOD_SERVE	IDLE
BACKHAND_0000	56.3%	4.2%	4.2%	33.3%	0%	2.1%	0%
BACKHAND_1111	16.7%	70%	0%	13.3%	0%	0%	0%
BAD_SERVE	0%	0%	83.3%	15.3%	0%	1.4%	0%
FOREHAND_0000	1.1%	0%	9.9%	84.6%	0%	4.4%	0%
FOREHAND_1111	0%	0%	0%	12.2%	87.8%	0%	0%
GOOD_SERVE	0.5%	0%	0.5%	0.5%	0%	97.3%	1.1%
IDLE	1.8%	0%	0%	0%	0%	3.5%	94.7%
F1 SCORE	0.65	0.79	0.83	0.75	0.94	0.97	0.96

Test Confusion Matrix

	BACKHAND_0000	BACKHAND_1111	BAD_SERVE	FOREHAND_0000	FOREHAND_1111	GOOD_SERVE	IDLE	UNCERTAIN
BACKHAND_0000	38.5%	4.6%	5.5%	2.8%	0%	3.7%	0%	45.0%
BACKHAND_1111	2.9%	73.5%	0%	0%	0%	2.9%	0%	20.6%
BAD_SERVE	0%	0%	33.8%	12.5%	0%	0%	5%	48.8%
FOREHAND_0000	0%	0%	2.8%	83.2%	0%	0.9%	0%	13.1%
FOREHAND_1111	0%	0%	0%	14.9%	61.7%	0%	0%	23.4%
GOOD_SERVE	0%	0%	0%	0.9%	0%	98.6%	0%	0.5%
IDLE	0%	0%	0%	0%	0%	11.5%	78.7%	9.8%
F1 SCORE	0.55	0.78	0.47	0.82	0.76	0.96	0.85	

We achieved an overall training accuracy of 86.8% and an overall test accuracy of 72.11%

Deployment

Our hardware target was the Arduino Nano 33 BLE Sense. We modified the deployment script to send the classification result to a laptop over BLE, which is received by a script running on the laptop which makes an API call to GPT with some hard-coded prompts depending on what the swing classification was, which is sent back over BLE to the Arduino. We further modified the Arduino script to display this text feedback on an OLED screen. Deployment also involved wiring up the screen to the Arduino as well as designing and 3d-printing an enclosure for the entire device.

Challenges and Lessons Learned

Our main challenges were in data collection and cleaning. One obvious limitation of our project is our sample size used for data, which was a consequence of limited time and indoor court availability due to weather. We only queried in total 6 individuals (2 advanced players, 2 intermediate, and 2 beginner players). We also did not obtain any data from left-handed players. Another issue was lost data. One of our Arduinos repeatedly stopped during data collection, resulting in lost motion data while the video kept recording. It turns out that this problem was caused by the player accidentally hitting the reset button on the Arduino. Similarly, if a player went out of frame in the video footage, that data was also considered lost.

Most detrimental to our model performance was the way we measured time in our motion data. We mistakenly saved the timestamp of when the motion data was received by the laptop, instead of the timestamps of when the data was collected by the Arduino. This resulted in what appeared to be an inconsistent sampling rate, as the delay of sending data from the Arduino to the laptop via BLE was variable. To make our data viable, we experimented with modifying the timestamps to reflect the original or some other constant sampling rate. This caused time in the motion data to be inconsistent with the actual time in our video data, leading to even more difficulty lining up the data. We are unsure how much our model performance is hurt by this discrepancy between the “sampling rate” in our training data and the true sampling rate at inference during deployment, as we observed that our deployed model performed significantly worse than either the training or test accuracy would suggest.

Another major limitation is that our device is only as good as our ability to correctly label tennis swings as “good” or “bad”. There may have also been bias between what one of us considers to be good or bad, so each swing may not have been labelled under exactly the same criteria either.

Our biggest learning from this project is how to balance detailed planning with keeping overarching goals in mind. For instance, we should have prioritized data collection from the outset, recognizing it as the most time-consuming task, instead of getting sidetracked by casing details or spending excessive time exploring alternative hardware like the Nordic Thingy 53. Additionally, we should have approached data collection with a clear strategy to streamline the cleaning process and continuously inspected the data as it was being gathered. Then we might have caught the sampling rate issue earlier. All in all, although our final product in the class demo was not as functional as we had hoped, the experience taught us a lot about the edge-AI problem solving process.

Link to other media:
https://drive.google.com/drive/folders/1rC9F7_qvLmHQCOSIB_YmuFlqXxmrylyB?usp=sharing