SWINGSCALE: A CROSS-PLATFORM MOBILE APP FOR REAL-TIME TENNIS FORM ANALYSIS USING MACHINE LEARNING

Bowen Li [1], Ang Li [2]

[1] Adlai E. Stevenson High School, 1 Stevenson Dr, Lincolnshire, IL 60069
[2] California State University Long Beach, 1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

In regions where access to professional tennis coaching is limited, aspiring athletes often find their potential capped by a lack of resources. SwingScale steps in as a game-changing mobile application, harnessing the precision of a custom-trained machine learning model to analyze tennis form right from the palm of the user's hand [14].

The SwingScale mobile application was developed utilizing the Flutter framework, enabling one cross-compatible source for both Android and iOS devices. The custom-trained machine learning, which serves as the key component for the SwingScale project, is a GradientBoostingClassifier model trained on a variety of different professional and amateur tennis form samples. The model boasts an average accuracy rate of 96% and consistently returns processed video analysis in no more than 12.5 times the video's duration.

Generally, throughout the project there were a few challenges encountered considering the advanced nature of the machine learning model and associated libraries and frameworks [1]. However, these challenges provided opportunities for reflection on the development process, and eventually were all overcome, and a final product able to be produced.

Reflecting on the challenges posed during development, inspired proper experiments to be conducted to analyze the performance of the challenge features. Multiple experiments were carried out and documented in this paper. The results demonstrate a consistent accuracy rate provided testing data, as well as a consistent analysis and response time back to the user.

Ultimately, the SwingScale mobile application offers a free, personalized tennis resource for individuals seeking to improve their game [3]. While other tools fall short in delivering tailored feedback, SwingScale empowers users with precise, consistent analysis of their recorded performance, thus providing classification and feedback customized to their unique skill level.

KEYWORDS

Mobile Coaching, Gradient Boosting, Sports AI, Tennis Feedback

David C. Wyld et al. (Eds): CRYPIS, CBIoT, CAIML, NLCA, NC, WiMo, ICAIT, ICDIPV, ITCSE – 2025
pp. 133-143, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151710

1. INTRODUCTION

In the game of tennis, a tennis player's serve is objectively a crucial skill the player must have to be a decent tennis player. However, players in undeveloped regions do not have access to the resources to develop their tennis serve and ultimately do not see their skill in tennis increase. Resources in underdeveloped countries are often not available to sport enthusiasts, as "the low level of participation in sports… are described as being due to the inadequate sport facilities, too little attention, and too little money allocated to sports and the development of physical culture" [4]. The lasting effect of the undeveloped tennis resources found in these regions is that potential great tennis players remain unfounded and unsupported, leaving a great chasm whose presence we stay unaware of and may never even discover. Additionally, the enjoyment and benefits of playing tennis as a recreational sport go unrecognized in communities that could benefit the most from such a presence. According to the International Journal of Physiology, Nutrition and Physical Education, "Sports not only improves physical health but also plays a significant role in improving mental health as there are many psychological benefits of sports" [5]. Therefore, it goes without saying that increasing participation and skill in a sport, tennis, could greatly improve the physical well-being of individuals in these underdeveloped regions.

Similar to the method proposed in this paper, other methodologies have been offered as a solution to the problem posed in this paper. Some of these methodologies include the Essential Tennis online library, the mobile application proposed in "Beyond the Coach..", and the Daily Tennis Lesson website [2]. The methods proposed from these sources of finding a solution to the lack of tennis resources in undeveloped areas, while making a genuine and helpful attempt, either lack in personalization or lack in completeness.

The first and last methods referenced earlier fall into the first category in that they lack personalization. While they are no doubt great sources of information and can provide a user with a great deal of tennis knowledge they may not already have, they lack personalized feedback on their own tennis ability. The SwingScale mobile app is able to analyze the players' form and provide personalized feedback on their own form.

This leaves the second referenced methodology, the mobile application proposed in the "Beyond the Coach.." paper. This mobile app, while it does provide technically personalized feedback, is not as elaborate as the SwingScale method. The mobile app attempts to compare the user's form to that of a professional tennis player, and returns the similarity percentage to the user as feedback. While this may be informative, it does not provide the user an analysis on the quality of their own form, and does not inform the user whether the shot was 'good' or 'bad.' The SwingScale mobile app provides the user feedback from an advanced machine learning model trained on a variety of quality data and is capable of grading a higher variety of tennis forms [6]. Additionally, the SwingScale app also provides LLM support, giving the user a more elaborate and clear response, including advice and tips for better performance.

As a solution to this problem, this paper proposes the SwingScale mobile application. The SwingScale project consists of three major components, all working together to provide an accessible and adequate application capable of providing a user with immediate and accurate feedback on their tennis serves. The SwingScale app is free to download on both of the major app stores, the Apple and Google Play stores, and serves as an easy-to-use frontend that anyone can manage. The behind-the-scenes process of the app utilizes a custom-trained GradientBoostingClassifier machine learning model, trained on tennis serves from a variety of different players, and is capable of classifying a good serve with a 96% accuracy rate. The SwingScale mobile application is available in any region and provides enhanced feedback utilizing an LLM trained to respond given the resulting output from the machine learning model

[7]. The resulting product is an easy-to-use app that gives custom-tailored feedback on a tennis player's serve by analyzing user-submitted videos. The combination of free usage and enhanced personal feedback makes the SwingScale app a competitive resource for tennis players and levels the playing field for tennis players who lack resources.

In this paper, multiple experiments were designed and performed to investigate and ensure the reliability and accuracy of the SwingScale application. These tests were performed in a manner to ensure an accurate and quick analysis to the user on their tennis shot performance.

The first experiment was performed to find the accuracy of the trained machine learning model. This model was designed and built to find the quality of a person's tennis form, provided a set of the player's landmarks from a video. The results of this experiment lead us to be confident in the model's ability to provide accurate results.

The second experiment carried out within this paper was intended to find and ensure a consistent processing time from the back-end server. Given the results of this experiment and the evident trend line in the data, we are led to conclude that the application's back-end server will remain consistent and provide the user base timely responses.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Flutter Challenges in SwingScale

One of the major challenges faced during the development of the SwingScale project was using the Flutter framework to build the mobile application. Flutter is a very cutting-edge framework and is constantly being updated or refactored. Because of this, changes to libraries and Flutter definitions were quite frequent, and often resulted in the changing of naming schemes or setups for certain objects. Additionally, the Flutter framework is quite expansive, and discovering new widgets to build certain items oftentimes requires extensive research in how to use them properly. Overall, the complexity and bleeding-edge nature of the Flutter framework proved to be an exciting challenge during the development of the app.

2.2. Overcoming YOLOv8 Integration Challenges

Another challenge we faced during the early development stages was configuring the machine learning model to properly identify the landmarks of required objects such as the tennis player, racket, ball, etc.. For the machine learning model, this project employed the use of the YoloV8 trained model, as it comes pre-trained to identify all of these objects out of the box. This was quite convenient for us, however, utilizing the YoloV8 model proved to be quite a challenge. This was due to the complexity of the Ultralytics library as well as the manipulation of the saved model file itself. Depending on the application, models can be saved in a variety of different file types. Identifying and utilizing the appropriate file type for our application was a whole process in itself.

2.3. Optimizing Model Accuracy with Gradient Boosting

Lastly, the challenge that we spent the most time on during the development of the overall application was getting a high accuracy score with the custom-trained machine learning model. During the experimentation process, we found that it takes a lot of data - and a variety of it - to

achieve a high accuracy score. Because of this, in order to get an accuracy score we were satisfied with, we were faced with acquiring more data to increase the size of our training dataset. Additionally, identifying what kind of machine learning model would provide the highest accuracy score given our dataset also proved to be quite challenging. Eventually, we settled on a Gradient Boost in Classifier as it proved to remain the most accurate out of all of the tests.

3. SOLUTION

Within the SwingScale project, the three main components responsible for the workflow include the mobile application interface, the computer vision tracking, and the machine learning algorithm for determining shot quality. To start, the user is prompted to upload a video of them performing a tennis shot and upon doing so their video is sent to the backend server for processing. When the backend server receives the video, the vision tracking model identifies and keeps track of the landmarks for the tennis ball, racket, and person. Once those landmarks have been broken down by the vision tracking model, the trained machine learning model receives them. The machine learning model then uses them to classify the quality of the tennis shot, grading them as ‘good’ or ‘bad.’ After the quality has been predicted, the classification is saved to the database where the user can eventually check from the mobile app. Additionally, the mobile app is capable of using ChatGPT prompting to analyze the landmarks and classifications to provide more context and helpful insight to the user [8]. To create the mobile application, we utilized the Flutter framework, as it provides a quick and convenient way to create a mobile application for both Android and iOS. To perform the vision tracking and landmark identification the YoloV8 model was employed for its accuracy. Finally, we used a Gradient Boosting Classifier trained in-house on our own dataset, partially created by us and partially sourced from the internet, to perform the classification on the landmark files.

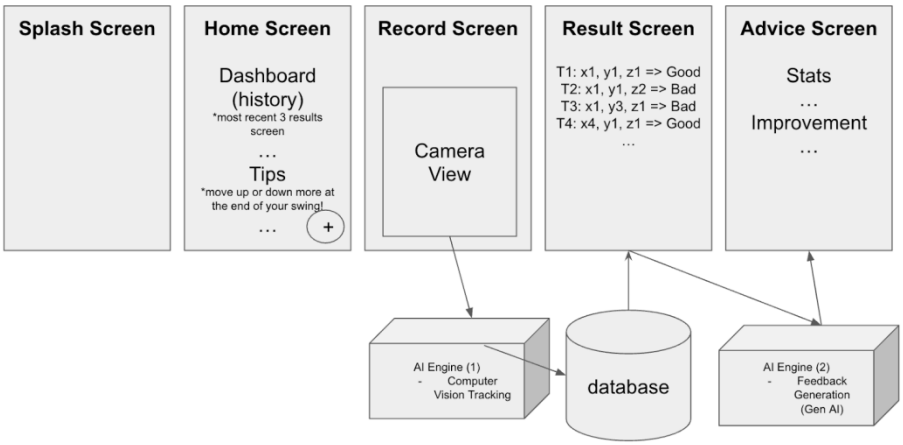


Figure 1. Overview of the solution

The SwingScale mobile app serves the purpose of allowing the user a quick and easy way of interacting with the backend server to receive their tennis video analyses. To build the mobile app, we utilized Flutter as it provides an efficient mobile application development framework.

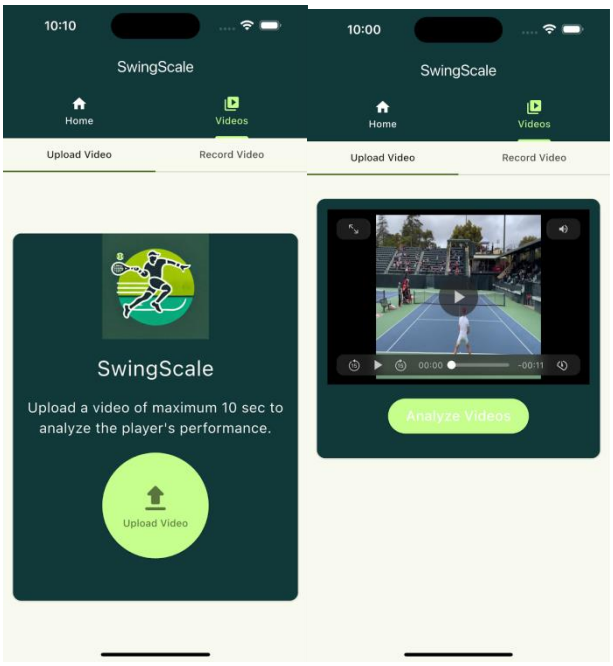


Figure 2. Screenshot of the videos

```dart
Future<void> sendVideoToServer(String filePath) async {
  try {
    // Update Firestore document
    await updateFirestoreDocument(_userId);

    // Create the custom HTTP client
    var customHttpClient = CustomHttpClient();

    // Create a MultipartRequest using the custom HTTP client
    var request = http.MultipartRequest('POST', Uri.parse('$url/analyze'))
      ..fields['userId'] = _userId
      ..files.add(await http.MultipartFile.fromPath('video', filePath));

    // Send the request using the custom HTTP client
    var response = await customHttpClient.send(request);

    if (response.statusCode == 200) {
      print('Video uploaded successfully');
    } else {
      print('Failed to upload video');
    }
  } catch (e) {
    print('Error: $e');
  }
}
```

Figure 3. Screenshot of code 1

In the code screenshot above, the sendVideoToServer function is defined. This function is, as the name suggests, responsible for taking the passed file path of the recorded video on the user's device and sending the file to the backend server. To do so, first, the user's Firestore document is updated to reflect that they are having a video processed by calling the updateFirestoreDocument function. Afterward, a custom HTTP client is initialized to send the video file. The request is then defined in the request variable, upon which it is then sent to the backend server utilizing the customHttpClient's send method. If the post request goes through fine, then the video has been uploaded successfully, if not, then an issue has likely occurred during the sending phase. While most of the Flutter code was for designing the look of the mobile application, this function plays a key role in enabling the communication between the user device and the back-end server.

The second major component within the SwingScale project is the computer vision tracking. The YOLO V8 model was implemented for tracking and detecting key features needed to determine the quality of the tennis serve [9]. These key features are kept track of in respective data frames and are later analyzed for determining shot quality.

Figure 4. Screenshot of score information

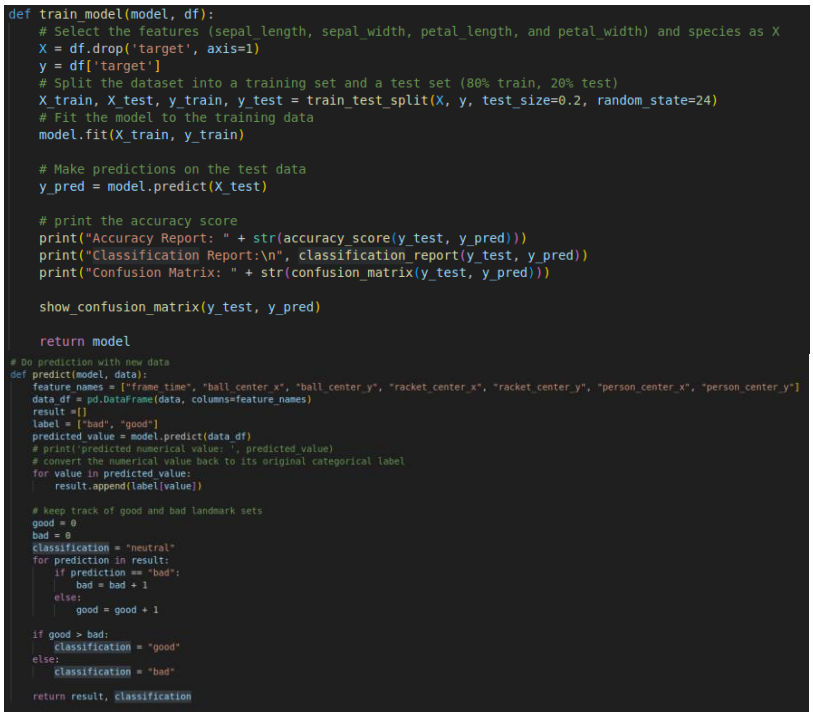
```python
def process_video(video_path, category):
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print(f"Error opening video: {video_path}")
        return None

    fps = cap.get(cv2.CAP_PROP_FPS)
    frames = []

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame_no = cap.get(cv2.CAP_PROP_POS_FRAMES)
        frame_time = round(frame_no / fps, 1)
        frame, h, w = resize_image(frame)
        ball_pos, frame = detect_ball(frame)
        person_pos, racket_pos, frame = detect_person_ball_racket(frame)

        if ball_pos is not None and racket_pos is not None and person_pos is not None:
            distance = calculate_distance(ball_pos, racket_pos)
            frames.append(
                {'frame_time': frame_time, 'distance': distance, 'ball_pos': ball_pos,
                 'racket_pos': racket_pos, 'person_pos': person_pos, 'category': category}
            )

    cap.release()
    return frames, h, w
```

Figure 5. Screenshot of code 2

In the above screenshot, the process_video function can be seen. The process_video function is the heart of the computer vision tracking process, in that it is responsible for taking the video and extracting the landmarks as frames. The function takes in two parameters: the video path and the category. The category is used for training purposes, in that the classification (1 if good, 0 if bad) can be passed to the function to include the category in the frames for dataset creation. If processing for a prediction and the category is unknown, any category can be passed, as the predict function ignores the category if it is present. The function processes the video in a while loop, frame by frame, and keeps track of all of the landmarks within the frame. It is important to note that the loop will not append the landmarks to the frame if one landmark is missing. This is

so that we do not include any data that is not an active tennis shot. The frames data frame and the height and width of the images are returned.

The final major component of the SwingScale project is the machine learning algorithm used for determining shot quality. The machine learning algorithm that is used is called the Gradient Boosting Classifier [10]. The Gradient Boosting Classifier trains on data from real-life tennis shots including various angles and players for enhanced performance evaluation.



```
def train_model(model, df):
    # Select the features (sepal_length, sepal_width, petal_length, and petal_width) and species as X
    X = df.drop('target', axis=1)
    y = df['target']
    # Split the dataset into a training set and a test set (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=24)
    # Fit the model to the training data
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)

    # print the accuracy score
    print("Accuracy Report: " + str(accuracy_score(y_test, y_pred)))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Confusion Matrix: " + str(confusion_matrix(y_test, y_pred)))

    show_confusion_matrix(y_test, y_pred)

    return model
# Do prediction with new data
def predict(model, data):
    feature_names = ["frame_time", "ball_center_x", "ball_center_y", "racket_center_x", "racket_center_y", "person_center_x", "person_center_y"]
    data_df = pd.DataFrame(data, columns=feature_names)
    result =[]
    label = ["bad", "good"]
    predicted_value = model.predict(data_df)
    # print('predicted numerical value: ', predicted_value)
    # convert the numerical value back to its original categorical label
    for value in predicted_value:
        result.append(label[value])

    # keep track of good and bad landmark sets
    good = 0
    bad = 0
    classification = "neutral"
    for prediction in result:
        if prediction == "bad":
            bad = bad + 1
        else:
            good = good + 1

    if good > bad:
        classification = "good"
    else:
        classification = "bad"

    return result, classification
```

Figure 6. Screenshot of code 3

In the above screenshots, both the train_model and predict functions from machine learning logic can be seen. The train_model function is quite simple, in that it all boils down to the ‘model.fit()’ call in which the passed data frame parameter is fit to the Gradient Boosting Classifier. Afterward, just for our analysis, we also created an accuracy and classification report as well as a confusion matrix. The predict function, however, is a bit more complex. The predict function receives the data to predict from the data parameter [11]. This data is a data frame processed using the previous process_video function from the computer vision logic on the user-submitted video. This data frame is predicted within the function using the ‘model.predict()’ method. Afterwards the total of ‘good’ and ‘bad’ classifications is found for all of the predicted landmarks. Whichever total is higher is the resulting classification for the user-submitted video.

4. EXPERIMENT

4.1. Experiment 1

One potential issue with the SwingScale application is the accuracy of the Gradient Boosting Classifier model. It is imperative that this model is accurate, as the app should be able to correctly classify the user’s tennis form.

```python
def show_confusion_matrix(y_test, y_pred):
    # Compute the confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Display the confusion matrix using a heatmap
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["bad", "good"], yticklabels=["bad", "good"])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

Figure 7. Code of experiment 1

To test the accuracy of our trained Gradient Boosting Classifier model we used a confusion matrix. The confusion matrix displays the count of total accurate classifications, which is a convenient method of directly seeing the results. To display the confusion matrix, we created the show_confusion_matrix function that can be seen above. The function utilizes the scikit-learn confusion_matrix function as well as matplotlib to create the confusion matrix as a graph. Additionally, we also implemented the get_test_results function to specifically test with data that was not included in the training dataset [12]. The creation of this function was in an attempt to avoid data biases.

```python
# Function to create tests with videos the model hasn't been trained on
def get_test_results(model, video_dir):
    try:
        # get the filenames in the directory
        filenames = os.listdir(video_dir)

        for filename in filenames:
            # create the video path
            video_path = os.path.join(video_dir, filename)
            # check that the path exists
            if os.path.exists(video_path):
                # Process the video
                result = fetch_dataframes(video_path, 0)    # category doesn't matter; it isn't used
                print(f"Result Frames:\n\n{result}")

                # Make predictions
                predictions, classification = predict(model, result)
                result['prediction'] = predictions

                # print(f"Result:\n{str(result)}\n")
                result.to_csv(f"data/result_{filename}.csv")

                print(f"Prediction for {filename}: {classification}")

    except Exception as error:
        print(f"unable to find prediction: {error}")
```

Figure 8. Code of experiment 1

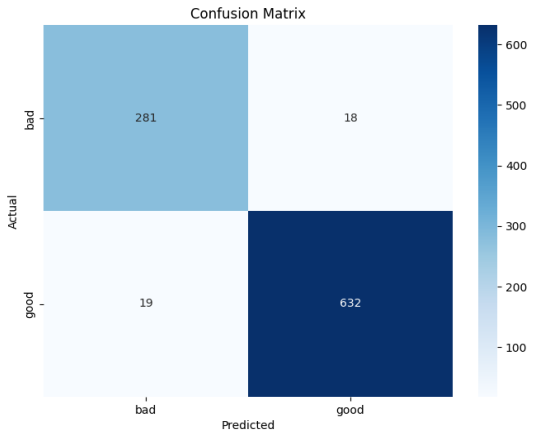


Figure 9. Figure of experiment 1

After training the model and performing the show_confusion_matrix function, we were able to gather our confusion as seen above. It is evident that the model is quite accurate, in that the majority of the predicted classifications are correct. Notably, we have found that the model is on average around 96% accurate ((281+632) / 950 = .96105) with the highest amount of accurate predictions being 632 accurate "good" classifications. The biggest surprise we encountered during the prolonged training phase was how much data it took to crawl closer to a 100% accuracy score. Previously in our training, we had a dataset size of around half of the current dataset size (950 total), and our accuracy score sat at around 92-93%. By doubling the data, we were only able to increase the accuracy score by around 3%. We believe the reason our accuracy score only increased this much is because model training and accuracy follow an exponential curve, where the more desirably accurate you want the model the more data it takes. We especially think this because of the complexity of the Gradient Boosting Classifier.

4.2. Experiment 2

A major factor in ensuring a quality user experience is the server response time. For our back-end server, it is important that the video processing and classification of the video is delivered in a timely fashion to the database for the user's viewing.

To test the proper functionality of the back-end server and to determine if there is a consistent response time, we will experiment to find the average response time given for videos of different lengths. To conduct the testing, the experimenters opened the server to track the processing and when the response would be returned. One experimenter will upload a video and the other will use a stopwatch such that when the response is returned, they will record the response time. The experimenters will record and log the video lengths and response times so that we will be able to create a slope line.

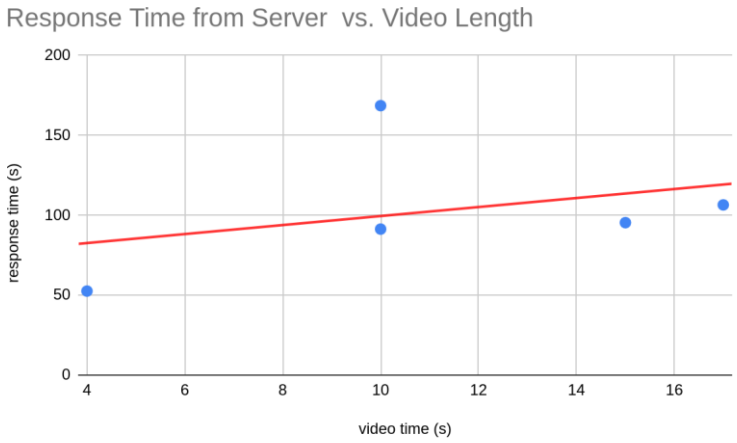


Figure 10. Figure of experiment 2

After performing the experiment and collecting the data, which can be seen displayed in the graph above, it is safe to conclude that the response time of our server is manageable. The results of our data demonstrate that there is a slope line with a value of m = 3.33, indicating that depending on the input video length, the length * 3.33 will roughly equate to the processing time. Additionally, we did find that out of our 5 trials, one proved to be an outlier. Specifically, one of our videos that had a length of 10 seconds ended up being processed for around 168.65 seconds. This is compared to the other video whose length was also 10 seconds which had a processing time of only 106.54 seconds. We believe that the reason for this difference is due to potential

video resolution differences. We have reason to believe that a higher quality of video may be impacting processing times by our model due to blatantly just having more data to process, as higher resolutions mean bigger images. Generally, after concluding our experiment, we are quite satisfied with the processing time multiplier as 3.33 is a quite modest value, given our experimentation and how accurate we know our model is.

5. RELATED WORK

One of the more popular services for providing free tennis support and resources is Essential Tennis, an online web resource that contains tennis tips, instructional blogs, and videos. Based on its popularity, it is safe to say that Essential Tennis does an adequate job at providing helpful resources for those looking to improve their tennis game. However, the Essential Tennis website is not capable of providing personalized feedback for each user's tennis serve form. The SwingScale mobile application is capable of providing tailored feedback for each user given an adequate recording of their tennis swing.

Another methodology that is quite similar to the SwingScale mobile application is the mobile detailed in the "Beyond the Coach.." research paper. The paper details a mobile application that is capable of taking a user video and comparing it to a selected video of a tennis professional hitting a very excellent serve. The mobile then compares the user's form in the video to the professionals and gives the user a rating of how similar their form is to the professionals. This mobile application, while capable of demonstrating differences in form, does not provide personalized feedback using an LLM or classification of whether their shot is good or bad. Rather, this app only specializes in comparing landmarks.

Daily Tennis Lesson is another online resource from a "Tennis Influencer" who goes by the name Brady [13]. The Daily Tennis Lesson website is filled with a lot of short to mid-range length video lessons, as well as other resources to help one improve their own tennis game. The obvious lacking component of the Daily Tennis Lesson website as a free tennis resource is personalized feedback. However, the website does leave methods to get into contact with 'Brady,' implying that one could get into contact for questions and perhaps even a video analysis. However, despite whether this could be feasible or not for a person dedicated to getting some feedback on their tennis form, the obvious implication is that it would take a while for Brady to respond, given that the email didn't go to spam. The SwingScale mobile application, as we demonstrated, is capable of giving accurate and insightful feedback in a timely manner, at the user's fingertips, whenever they need it.

6. CONCLUSIONS

Looking back on the SwingScale project, while it is clear to see that it is useful, the app can definitely stand to be improved upon. One area in which the project can be improved is the nature of the machine learning model. Currently, the GradientBoostingClassifier is mainly only a classification model and is limited in the information that it provides. If the model instead provided a percentage from bad to good form, the model could prove to be much more insightful for users. Additionally, the LLM feedback currently giving users advice on their landmarks does not have extensive knowledge on what good landmarks look like and is left to conclude what 'good' or 'bad' landmarks are during its generation. To resolve this issue, we could finetune a model specifically to give it a better idea of what landmarks are good or bad. To achieve this, we would need a dataset of landmarks and a corresponding description of what the player is doing and whether the serve was good or bad to provide to the LLM model [15].

To conclude, the SwingScale app provides a service to those who lack the resources to advance their tennis game but maintain the will and drive to do so. We would like to take a moment to thank the reader for recognizing the significance of this research, and we would appreciate it if you would share this application as we wish to benefit as many people as we can.

REFERENCES

[1] Sudhir, G., John Chung-Mong Lee, and Anil K. Jain. "Automatic classification of tennis video for high-level content-based retrieval." Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database. IEEE, 1998.

[2] Hao, Jiawen, and Huijun Hu. "Beyond the coach: Exploring the efficacy of a machine learning application for improving tennis players' performance." CS & IT Conference Proceedings. Vol. 13. No. 9. CS & IT Conference Proceedings, 2023.

[3] Kawakami, Ryo, et al. "Characteristics of physical activity during beginner-level group tennis lessons and the effect daily activity." Scientific Reports 14.1 (2024): 249.

[4] Andreff, Wladimir. "The correlation between economic underdevelopment and sport." European sport management quarterly 1.4 (2001): 251-279.

[5] Hiremath, Chandrakanta. "Impact of sports on mental health." International Journal of Physiology, Nutrition and Physical Education 1 (2019): 14-18.

[6] Kinari, Safira Adine, et al. "A Study of Guide Documentation for Introductory Flutter Programming Learning with Exercises." 2025 1st International Conference on Consumer Technology (ICCT-Pacific). IEEE, 2025.

[7] Kramer, Oliver. "Scikit-learn." Machine learning for evolution strategies. Cham: Springer International Publishing, 2016. 45-53.

[8] Mufid, Mohammad Robihul, et al. "Design an mvc model using python for flask framework development." 2019 International Electronics Symposium (IES). IEEE, 2019.

[9] Khan, Md Saikat Islam, et al. "Water quality prediction and classification based on principal component regression and gradient boosting classifier approach." Journal of King Saud University-Computer and Information Sciences 34.8 (2022): 4773-4781.

[10] Yim, Aldrin, Claire Chung, and Allen Yu. Matplotlib for Python Developers: Effective techniques for data visualization with Python. Packt Publishing Ltd, 2018.

[11] Auger, Tom, and Emma Saroyan. "Overview of the OpenAI APIs." Generative AI for Web Development: Building Web Applications Powered by OpenAI APIs and Next. js. Berkeley, CA: Apress, 2024. 87-116.

[12] Özel, Muhammed Abdullah, and Mehmet Yasin Gül. "Ensuring product detection and product counting on the assembly line using deep learning (YOLOv11)." International Journal of Multidisciplinary Studies and Innovative Technologies 9.1: 53-58.

[13] Kumar, Ashok. Mastering Firebase for Android Development: Build real-time, scalable, and cloud-enabled Android apps with Firebase. Packt Publishing Ltd, 2018.

[14] Schefer-Wenzl, Sigrid, and Igor Miladinovic. "Game changing mobile learning based method mix for teaching software development." Proceedings of the 16th World Conference on Mobile and Contextual Learning. 2017.

[15] Alberts, Ian L., et al. "Large language models (LLM) and ChatGPT: what will the impact on nuclear medicine be?." European journal of nuclear medicine and molecular imaging 50.6 (2023): 1549-1552.