Project_Team01_report

- **Introduction** –

  With the rapid development of data analysis, more and more of them have been applied to the sport science field, especially the competitive one to boost the performance of players, predict the result of competitions, etc. Inspired by the shuttleNet framework for badminton shot type predicting (Wang, *et al.* 2021) and Alphafold2 (Jumper, *et al.* 2021) , we decided to attend the coachAI badminton challenge to make more accurate predictions about turn based stroke in badminton rallies by fusing two framework mentioned above. The challenge is mainly about predicting the shot type and the placement of the badminton with past stroke sequences.

- **Literature Review/Related work** –

  The transformative impact of data-driven decision-making on sports analytics has unlocked vast potential for enhancing strategies, performance analysis, and viewer engagement. Among various sports, turn-based games like badminton present intriguing challenges due to their inherent variability and complexity. A pioneering study titled "ShuttleNet: Position-aware Fusion of Rally Progress and Player Styles for Stroke Forecasting in Badminton" by Wang et al., (2021) delved into an unexplored domain of badminton analytics, aiming to predict future strokes—types and locations—based on past stroke sequences. This predictive prowess promises significant improvements in coaching strategies and the spectator experience.

  The ShuttleNet model addressed the unique challenges of stroke prediction in badminton with a novel design. Conventional sequence prediction models fall short due to badminton's distinctive rally characteristics: mixed-sequence nature, the requirement for multiple output predictions, and the dependence on player style and current rally situation. ShuttleNet deftly navigated these issues through a combination of two encoder-decoder extractors to model rally progress and player styles, alongside a fusion network that considered dependencies between the rally progress and player styles at each stroke. However, the swift evolution of AI and machine learning landscapes implies untapped potential for further enhancing prediction accuracy and efficiency.

In this paper, we propose augmenting the ShuttleNet model by incorporating the AlphaFold2 (Jumper et al., 2021) architecture, known for its groundbreaking contributions to protein structure prediction, and the DINO (Caron et al., 2021) framework, recognized for its proficiency in extracting valuable representations from limited data.

AlphaFold2's remarkable success in protein structure prediction surpasses all previous models in terms of the Z-score sum, reducing prediction error down to the atomic scale. This success is attributed to their emphasis on learning representative features expressed as multiple sequence alignment (MSA) and pair representations. MSA representation, extracted from a genetic database, represents evolutionary information about the amino acid sequence, while pair representation provides a tabular expression of the relationship between each amino acid pair within the sequence, reflecting thermodynamic and stereochemical information. With these representations, AlphaFold2 introduced Evoformer, a primarily gated-encoder (transformer)-based architecture, to update representation elements. After three rounds of updates—termed recycling—the representations were fed into the structure module, the model's decoder, to generate the final structure prediction.

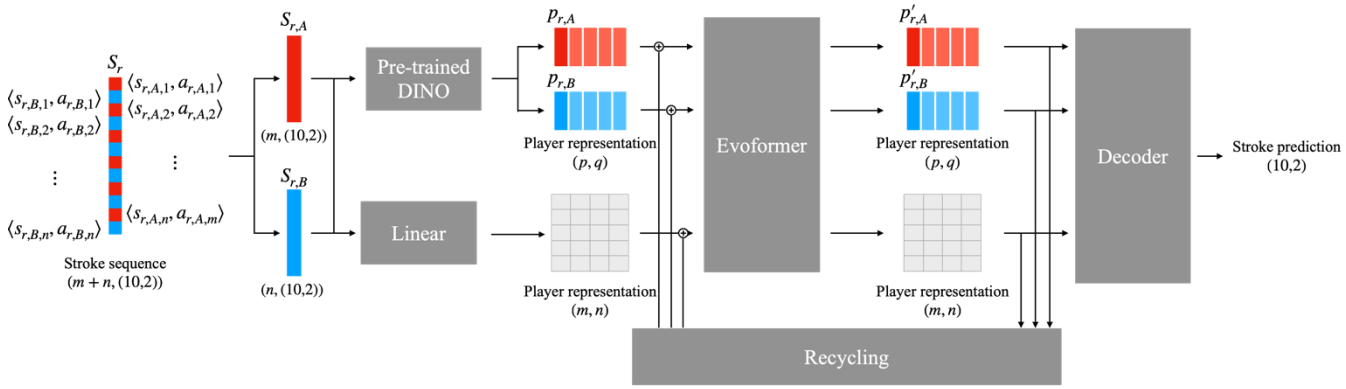
Inspired by AlphaFold2, we identified two analogies between AlphaFold2 and our task: 1) the MSA representation is analogous to a single player's behavioral pattern (player representation), and 2) the pair representation is akin to the correlation of each stroke of two players throughout the rally (rally representation). Therefore, we plan to substitute MSA and pair representations with player and rally representations in our architecture, leveraging AlphaFold2's ablation studies.

Nevertheless, the training processes of protein prediction and stroke forecasting have substantial differences. The most notable difference lies in the training data volume. While AlphaFold2 was trained with a self-distilled dataset and a protein database containing almost half a million training data, our model can only utilize a maximum of 33,612 training data. Moreover, MSA representation initialization is based on similar sequences due to amino

acid sequence similarity. In contrast, player representation initialization can be based on not only previous stroke similarity but also rallies played by the same player. Given these differences, we may need to revise the initialization workflow using contrastive learning techniques. To distill meaningful and useful representations, we have chosen the DINO (Caron et al., 2021) framework due to its beneficial properties, including no need for negative samples (similar to BYOL (Grill et al., 2020)), centering and sharpening operations to avoid learning trivial or uninformative representations, and its usage of the vision transformer.

This paper, therefore, extends the ShuttleNet model by infusing the architecture with elements from AlphaFold2 and DINO. The aim is to enhance the ability of the model to understand and represent player behavior and rally progress in badminton. The introduction of the AlphaFold2-inspired approach enables a more nuanced sequence analysis, drawing parallels between protein structure prediction and badminton stroke prediction. We expect this approach to improve the accuracy of our model by effectively capturing the dynamics of the game. To address the limited training data available for badminton stroke prediction, we adapt the DINO framework's principles. DINO's ability to learn from limited data and avoid trivial representations make it an ideal choice to refine our model's training process. This inclusion is intended to make the model more robust and effective in predicting strokes under varying circumstances and player styles.

We believe our proposal of combining the ShuttleNet model with elements from AlphaFold2 and DINO represents a promising direction in sports analytics, particularly in stroke forecasting for turn-based games like badminton. By leveraging the strengths of these cutting-edge technologies, we aim to push the boundaries of predictive analytics in sports, potentially revolutionizing game strategies, player performance analysis, and spectator engagement. In the sections that follow, we will elaborate on the specific methodologies used to integrate these elements into the ShuttleNet model, present the results of our experiments, and discuss the potential implications of our findings for the broader field of sports analytics.



- **Dataset** –

  In this study, we utilize the same dataset as used in the ShuttleNet model, generously provided by the challenge organizers. The dataset is bifurcated into two primary sections, each catering to different aspects of the game. The first section comprises 33,612 turns of detailed, turn-based information. This portion of the dataset incorporates 26 columns encompassing a myriad of variables such as the current round score of players, the players' locations, and so forth. These variables offer deep insights into the game's dynamics at a granular level, serving as a rich resource for our model to learn from. The second section is centered around match-level information, including variables like match duration, the match's winner, among other elements. This section comprises 58 rallies, with each rally further broken down into six columns of data. This data assists our model in understanding the broader context of each match, providing a comprehensive perspective on the game.

  To ensure the versatility of our model and to avoid overfitting, we divided the dataset into training, validation, and testing subsets. The data is apportioned in a ratio of 9:0.5:0.5, respectively. Furthermore, to enhance the robustness of our model and improve the reliability of our results, we have employed a 5-fold cross-validation strategy. This approach guarantees that our model is trained and evaluated on different subsets of the data, promoting a generalized understanding and more accurate stroke prediction performance.

- **Baseline** –

In this study, we proposed a Multilayer Perceptron (MLP) model and ShuttleNet (Wang *et al*., 2021) as a baseline method to compare with other models.

According to our experimental results, our MLP model achieved the score of 4.75 on the selected evaluation metric.

In addition to our MLP model, we also used ShuttleNet model as another baseline method. After training and testing the ShuttleNet model with the same dataset, we obtained a score of 2.8. The score indicates that the ShuttleNet model achieved a relatively higher level of accuracy compared to our MLP model baseline.

The following is loss function we used in this project, which is same as the metric in competition:

$$loss = AVG(CE + MAE)$$

- **Main Approach** –

We commence our methodology by detailing the dataset employed and the associated data preprocessing steps. Following this, drawing upon the resemblances with AlphaFold2, we elucidate the initialization of our unique player and rally representations. Subsequently, we delve into the intricacies of the Evoformer and the recycling trick, which facilitates the update of these representations.

The discussion then segues into the decoder, which transforms these representations into stroke forecasts. Concurrently, we elaborate on our loss engineering approach, devised to optimize our model's performance. Finally, we outline the evaluation metrics and techniques employed to assess and validate the efficacy of our models. In each segment, we strive to elucidate the underpinning logic and implementation details to comprehensively understand our methodology.

The historic rallies of badminton matches are denoted as $R = \{S_r, P_r\}_{r=1}^{|R|}$, where S_r stands for the r-th stroke sequence composed of $\left(\langle s_1, a_1\rangle, \ldots, \langle s_{|S_r|}, a_{|S_r|}\rangle\right)$ and P_r stands for the r-th player sequence. For any i-th stroke, s_i represents the shot type, $a_i = \langle x_i, y_i\rangle \in \mathbb{R}^2$ represents the

coordinate of the shuttle destination, and p_i represents the player who hits the shuttle. With the formulation of our data, our task is to predict the shot types and area coordinates of the future n strokes ($\langle s_i, a_i \rangle_{i=\tau+1}^{\tau+n}$) given the observed τ strokes ($\langle s_i, a_i \rangle_{i=1}^{\tau}$).

Player Representation

The DINO is used to initialize player representation concerning insufficient training data. The distilled player representation can be formulated as follows:

$$p_{i,A} = g_{\theta_s}(\langle s_{i,A}, a_{i,A} \rangle), \forall i \in [1, |S_r|]$$

, where $p_{i,A}$ is the player representation of player A for the i-th stroke sequence, $\langle s_{i,A}, a_{i,A} \rangle$ is the stroke sequence of player A for the i-th stroke sequence, and g_{θ_s} is the well-trained student network in DINO. After the transformation from stroke sequence to player representation, there're two ways to collect the representation to load into the Evoformer. One approach is to collect the representations with the lowest Euclidean distance from the query representation, and the other is to collect the other strokes played by the same players. Which way is more appropriate should be tested later. For convenience, we denote the collected player representation as p_q.

Rally Representation

Unlike the initialization of player representation pre-trained before loading into Evoformer, the initialization of rally representation would be trained with Evoformer. The rally representation would be formulated as follows:

$$r_{q,j,k} = \text{Linear}_{rally}(\langle s_{i,A,j}, a_{i,A,j} \rangle, \langle s_{i,B,k}, a_{i,B,k} \rangle)$$

, where $r_{q,j,k}$ is the (j,k) element of query rally representation, Linear_{rally} is the simple multiple layer perceptron for the initialization of rally representation, $\langle s_{i,A,j}, a_{i,A,j} \rangle$ is the j-th stroke of player A in the i-th stroke sequence, and $\langle s_{i,B,k}, a_{i,B,k} \rangle$ is the k-th stroke of player B in the i-th stroke sequence.

Evoformer Architecture

The Evoformer, a key component of our model, has been adopted from the groundbreaking AlphaFold2 structure. It is a transformer-based architecture that excels at capturing the complex dynamics and evolving patterns inherent in badminton games. The Evoformer comprises a self-attention mechanism, a feed-forward network, and layer normalization processes, all working together to iteratively update our player and rally representations. However, to adjust the architecture to our unique context, we interpret these mechanisms in the context of our player and rally representations instead of MSA and pair representations as in AlphaFold2. The self-attention mechanism acts as the cornerstone of the Evoformer. It identifies and represents dependencies across various elements of our representations. In our adaptation, we incorporate a two-step process that mirrors AlphaFold2's row-wise and column-wise attention mechanisms:

1. Player-wise attention (analogous to row-wise attention): Each player's behavior during a rally is treated as an independent sequence. This attention step works similarly to the standard self-attention mechanism used in transformer models, treating each sequence as an independent entity. The output of the player-wise attention is the updated player representation that reflects the dependencies within each player's actions.

2. Rally-wise attention (analogous to column-wise attention): This mechanism captures inter-sequence dependencies, or dependencies across different player sequences in a rally. The output of the rally-wise attention is a further updated player representation that reflects dependencies across different player actions.

Next, the feed-forward network refines our representations further. Comprising two successive linear transformations separated by a ReLU activation function, it allows our Evoformer to capture non-linear dependencies within the data. Layer normalization is applied after both the attention and feed-forward mechanisms. It maintains a consistent scale for the outputs across different layers, which is critical for stabilizing the learning process, particularly during the recycling phase when the representations are iteratively updated. Our player and rally representations are input into the

Evoformer and undergo iterative updates in a process referred to as 'recycling'. This involves three rounds of updates within the Evoformer, each allowing the model to capture deeper, more complex relationships in the data and thereby enhancing the accuracy of stroke prediction.

Finally, the pair representation update process used in AlphaFold2 is mirrored in our model by updating the interaction representations, which capture the pairwise interactions between different player strokes in a rally. An outer product of the updated player representation is calculated, generating an intermediate interaction representation. This representation is then concatenated with the original interaction representation and goes through a self-attention mechanism, updating it to capture complex interactions between different player strokes.

Recycling Representations

The concept of recycling representations, borrowed from the AlphaFold2 architecture, plays a pivotal role in our model's methodology. Essentially, recycling refers to the iterative update of our representations—the player and rally representations—within the Evoformer. This technique allows for a more thorough and nuanced understanding of the game's dynamics, capturing the complexity of badminton rallies and player styles. In the context of our model, recycling is implemented as a three-stage process. The player and rally representations are first fed into the Evoformer, which serves as the core transformer-based architecture in our model. The Evoformer is designed to iteratively refine and update these representations.

In the initial stage, the Evoformer takes the raw player and rally representations as inputs. These inputs undergo an update process that relies on the transformer's ability to capture dependencies and correlations within the input data. The output of this first stage then serves as the input for the second stage. In the second stage, the Evoformer processes the updated representations from the first stage. It further refines these representations, capturing deeper and more complex relationships in the input data. Once

again, this stage's output serves as the final stage's input. During the final stage, the Evoformer processes the twice-updated representations. This third update allows the model to reach the depth effect of three times the architecture's depth but conserves memory due to the shared weights in the Evoformer. It ensures that the most intricate aspects of the rally progress and player styles are incorporated into the representations.

Upon completion of the recycling process, the thrice-updated representations are loaded into the structure module, serving as the decoder of our model. This decoder then translates these enriched representations into stroke forecasts. Through the recycling of representations, our model is endowed with a deeper understanding of the underlying patterns in badminton, thereby enhancing the accuracy of its stroke predictions.

Decoder Architecture

In the final stage of our model's stroke prediction pipeline, we employ a transformer-based decoder architecture. This architecture is adept at handling the complexities and dependencies within the player and rally representations that have been refined through the Evoformer. The transformer decoder intakes the processed player and rally representations, operating on both simultaneously. Each representation undergoes a series of transformations within the transformer decoder, which employs self-attention mechanisms to model interdependencies across different positions in the sequence. This approach permits the model to consider the broader context and intricate relationships within the stroke sequences. Following the self-attention layers, a feed-forward neural network captures any remaining non-linear dependencies within the data, ensuring a comprehensive understanding of the player and rally dynamics.

The decoder outputs two distinct predictions: the type of the next stroke and its location on the court. To do this, the transformer decoder features separate output layers for each prediction type. The stroke type output layer employs a softmax activation function to generate a probability distribution over

potential stroke types. In contrast, the stroke location output layer utilizes a linear activation function, yielding continuous coordinates for the predicted stroke's landing point on the court. We use a specialized loss function in our model to balance the importance of accurate stroke type and location predictions. During the training process, the model's parameters are adjusted to minimize this loss, thereby improving the model's predictive accuracy over time.

- **Evaluation Metric** –

Let $R = \{S_r, P_r\}_{r=1}^{|R|}$ denote historical rallies of badminton matches, where the r-th rally is composed of a stroke sequence with type-area pairs $S_r = (\langle s_r, a_1\rangle, \ldots \langle s_{|S_r|}, a_{|S_r|}\rangle)$ and a player sequence $P_r = (p_1 \ldots p_{|S_r|})$. At the i-th stroke, S_i represents the shot type, $a_i = \langle x_i, y_i\rangle \in \mathbb{R}^2$, are the coordinates of the shuttle destinations, and p_i is the player who hits the shuttle. We denote Player A as the served player and Player B as the other for each rally in this track. For instance, given a singles rally between Player A and Player B, p_r may become (A,B …, A,B). We formulate the problem of stroke forecasting as follows. For each rally, given the observed τ strokes $(\langle s_i, a_i\rangle)_{i=1}^{au}$ with players $(p_i)_{i=1}^{\tau}$, the goal is to predict the future strokes including shot types and area coordinates for the next n steps, i.e., $(\langle s_i, a_i\rangle)_{i=\tau+1}^{\tau+n}$

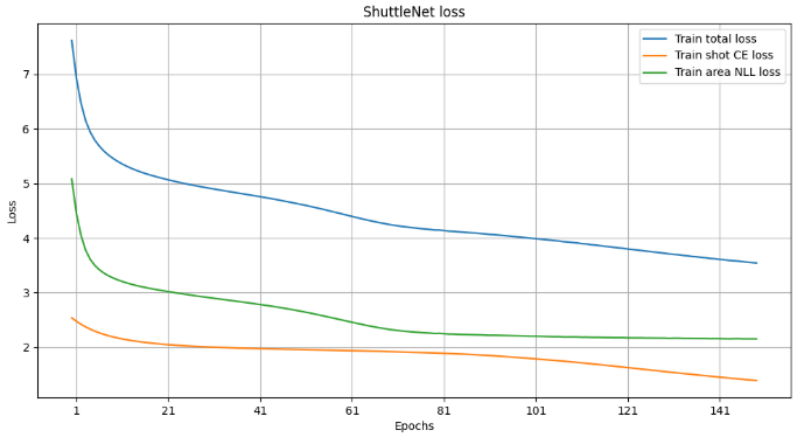
shot type: Cross Entropy (CE)
landing_x, landing_y: Mean Absolute Error (MAE)
$Score = \min(l_1, l_2, \ldots., l_6)$

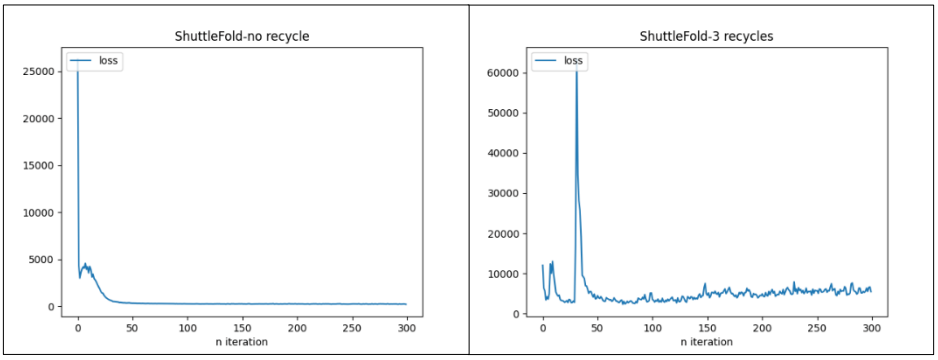
$$l_i = AVG(CE + MAE) = \frac{\sum_{r=1}^{|R|}\sum_{n=\tau+1}^{|r|}[|x_n - \widehat{x_n}| + |y_n - \widehat{y_n}|]}{|R|\cdot(|r| - \tau)}$$

- **Results & Analysis** –

As abovementioned, we deemed recycle trick as the critical trick in AlphaFold2 for protein structure prediction, but the process of recycling to highly degree decreased the throughput of the model. In our task, we're similarly interested in whether recycling make impact in our model, especially in the shot prediction task. Hence, we conduct a trial with and without recycling and plot the learning curve (loss) for comparison.

As shown in our figures, the recycle trick is an expensive overhead here in our model and makes the model harder to train. It's contradictory to our previous understanding about recycling, because in AlphaFold2, the author figured out that the recycling trick can indirectly deepen the model but train on lesser number of parameters. Here we have a hypothesis of this poor performance, the more we recycle our model, the more the gradient vanish from the perspective of first half part of our model, subsequently the model only fine-tuning the later part of our model and easily converge on relatively non-ideal results.





- **Future Work** –

  DINO wasn't used in this model for player representation due to time issue, and we would like to implement it in near future to improve model performance. Due to the massive architecture of our model, the ablation study should be done to clarify which part or trick mostly contribute to the performance; however, it's quite time-consuming and redundant for competition circumstances, so we didn't implement this time and it would be prospective to be done in the future.

- **Code** –

https://github.com/jimchen1551/ShuttleFold?fbclid=IwAR07jmFbEu_9LrcC6_CU6OAwXBUB6_Pvge-6QEHibixNYjzyngAno411qCc

- **Contribution of each member** –

  10801128 陳俊鴻 40%
  110612117 張仲瑜 30%
  111101016 崔芸禎 15%
  111101017 俞丞訓 15%

- **References** –

[Wang *et al*., 2021] Wei-Yao Wang, Hong-Han Shuai, Kai-Shiang Chang, and Wen-Chih Peng. Highly accurate protein structure prediction with AlphaFold. *AAAI*, 2022.

[Jumper *et al*., 2021] John Jumper, *et al*. Highly accurate protein structure prediction with AlphaFold. *Nature, 596(7873), 583–589*, 15 July 2021.

[Caron *et al*., 2021] Mathilde Caron *et al*. Emerging Properties in Self-Supervised Vision Transformers. *ICCV*, 2021.

[Grill et al., 2020] J. B. Grill *et al*. Bootstrap Your Own Latent A New Approach to Self-Supervised Learning. *NIPS*, 2020.