Learning the Neural Network to Play Blood Bowl

Supervisor: Krzysztof Hryniów, PhD

Wiktor Wołek
Faculty of Electrical Engineering
Warsaw University of Technology
Warsaw, Poland
wiktor.wolek.stud@pw.edu.pl

Bartosz Nowicki
Faculty of Electrical Engineering
Warsaw University of Technology
Warsaw, Poland
bartosz.nowicki4.stud@pw.edu.pl

Karol Kociołek
Faculty of Electrical Engineering
Warsaw University of Technology
Warsaw, Poland
karol.kociolek.stud@pw.edu.pl

***Abstract*—This paper presents a comparative study of different neural network architectures trained using behavioral learning in the strategic board game Blood Bowl. The game's complexity, driven by its large branching factor and inherent randomness, presents a significant challenge for artificial intelligence (AI). Traditional AI approaches, such as scripted and search-based methods, have struggled to achieve human-level performance. This study examines how various deep learning models process decision data, adapt strategies, and handle uncertainty in game-play. Our methodology involves training models with a custom preprocessing pipeline that extracts valid game states and actions from replays of scripted solutions and tournament self-play. Performance is evaluated by competing against existing AI solutions, using metrics such as win rates, move efficiency, and strategic accuracy. The results highlight the comparative strengths and weaknesses of each architecture, providing insights into their effectiveness for reinforcement learning-based agents in complex decision-making environments like Blood Bowl.**

***Index Terms*—Reinforcement Learning, Imitation Learning, Blood Bowl, AI, OpenAI Gym**

I. Introduction

Board games have long been regarded as activities requiring players to think strategically and make quick, accurate decisions. Not all board games share the same complexity; for instance, games like chess and Go have relatively well-studied branching factors and deterministic mechanics, making them more tractable for search-based algorithms [1]. However, Blood Bowl—the game of focus in this paper—presents unique challenges due to its massive branching factor and considerable randomness.

The primary goal of this work is to examine whether advanced machine learning techniques, particularly behavioral cloning and reinforcement learning, can overcome the difficulties posed by such a high-dimensional, stochastic environment. Traditional scripted approaches and pure forward-search methods have been inadequate in capturing the nuance and unpredictability of Blood Bowl. This paper posits that an imitation-learning approach, supplemented by reinforcement learning, may enable the training of a competitive AI agent.

We provide a comparative study of multiple neural network architectures trained via Behavioral Cloning (BC) on data extracted from scripted replays. We investigate the impact of both network design and data collection strategies on the overall performance of the agents.

II. Blood Bowl and the BotBowl (FFAI) Framework

Blood Bowl is a turn-based board game that blends elements of American football and fantasy war gaming. Two teams of players alternate turns, with the aim of scoring touchdowns (TD) by moving the ball into the opposing end zone. A defining characteristic of Blood Bowl is its high level of randomness, introduced through dice rolls for actions like movement, blocking, and ball handling. This stochastic aspect, combined with the vast decision space on each turn, significantly increases the complexity of designing competitive AI agents which have to balance short-term tactical choices with long-term strategic planning.

The BotBowl framework (Figure 1), also referred to as the *FFAI* environment, has been developed to facilitate AI research and competition in Blood Bowl-like scenarios [3]. BotBowl provides:

- A simulator for the core rules of Blood Bowl, including all randomness and event handling.
- An interface for controlling agents, receiving state information, and executing actions.
- Standardized match formats to evaluate performance and compete against other bots.
- Tools for logging, debugging, and visualizing matches to aid in AI development and analysis.

This environment streamlines data collection and bot evaluation, ensuring that researchers can focus on strategy development, state representation, and policy learning without manually simulating complex game mechanics.

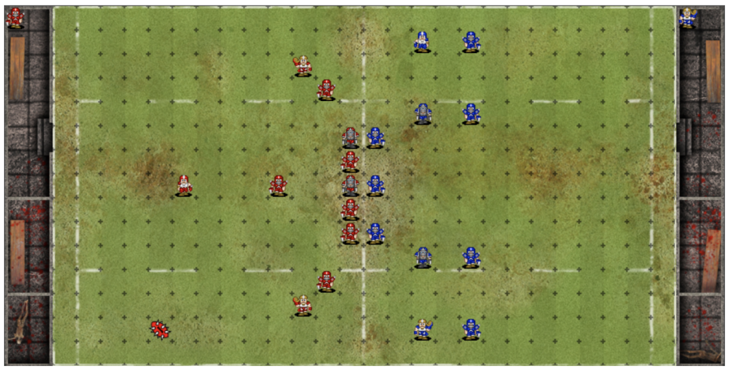

Fig. 1. The game board in FFAI after both teams have set up

III. Literature Review

Blood Bowl's complexity far exceeds that of many other board games, primarily due to its extensive branching factor and stochastic elements. Justesen et al. (2019) highlighted the unique challenge posed by Blood Bowl, noting an average turn-wise branching factor of approximately 10^{50}, compared to 30 for chess and 300 for Go [3].

Furthermore, Humeau et al. (2021) proposed a stochastic model of Blood Bowl's main phase using a Markov Decision Process. Their multi-layered framework integrates expert knowledge to efficiently handle the game's vast state space and randomness, providing a solid basis for advanced planning algorithms [9].

Earlier research demonstrated that combining machine learning techniques with scripted knowledge could outperform purely scripted bots. Pezzotti (2021) [4] found that a hybrid approach using both imitation learning and reinforcement learning offered decent performance compared to non-scripted methods.

In addition to these studies, recent advances in real-time strategy games provide valuable insights applicable to Blood Bowl. For example, the work by Vinyals et al. (2019) [8] demonstrated that a hybrid learning strategy can produce agents that perform at a Grandmaster level in StarCraft II.

These results underscore the viability of using advanced machine learning and hybrid methods for Blood Bowl. However, the sheer complexity and randomness of the environment pose critical challenges, especially for standard algorithms like pure reinforcement learning or naive search-based approaches.

IV. Methodology

Our research focuses on training AI agents to play BotBowl (Blood Bowl) effectively. Early experiments revealed that straightforward approaches, such as forward-search methods (e.g., Monte Carlo Tree Search), struggled with the exponential explosion of the state-action space. Similarly, pure reinforcement learning methods often failed to converge on robust strategies, largely due to sparse and delayed rewards in a highly stochastic setting.

A. Behavioral Cloning Approach

A more promising direction emerged through Behavioral Cloning (BC), where agents learn from demonstrations provided by stronger, but not necessarily optimal, scripted or rule-based bots. Although BC quickly led to agents capable of executing basic strategies, it inherited the performance ceiling of the source demonstrations, limiting any improvements beyond the demonstrated policy.

We also attempted fine-tuning these BC-trained agents with reinforcement learning (Figure 2), aiming to surpass the limitations of the original demonstrations. However, these efforts typically faced instability and inconsistency, presumably due to the complexities of exploring such a large, random environment.

Given these constraints, we focus on:

1) Collecting high-quality demonstration data from scripted bots.

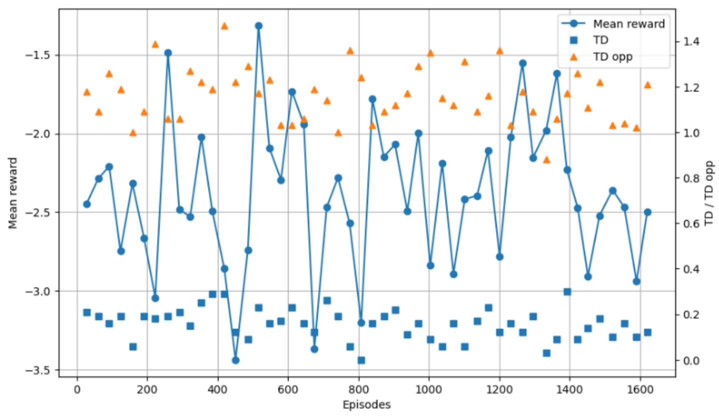

Fig. 2. Example reinforcement learning performance with behavioral cloning step

2) Testing how distinct neural network architectures—particularly those that incorporate convolution, residual connections, attention mechanisms, and transformers—affect performance when trained via BC.
3) We compare our agents against a standard scripted bot, a random bot, and variants of our own models.

V. Behavioral Cloning Performance

Here, we detail the neural network architectures and examine their performance under a unified training protocol. The evaluation metric centers on match outcomes when our trained models compete against standard baselines: scripted and random bots.

A. Architectures

We evaluated three network architectures, each trained for five epochs using data from 1500 games, during which the moves of a scripted bot were recorded as it competed against various opponents:

- **Network1**: This architecture draws inspiration from AlphaStar-like [8] models. A convolutional neural network processes spatial data using residual blocks and attention modules to extract high-level board representations. Non-spatial input features are embedded and combined with a special classification token, then passed through a Transformer block to capture complex dependencies.
- **Network2**: Influenced by the MimicBot architecture [4], this model integrates convolutional feature extraction with channel-wise attention to effectively merge spatial and non-spatial information. Each residual block consists of three convolutional layers with LeakyReLU activations and shortcut connections, promoting stable and deep feature learning. Channel attention mechanisms reweigh the extracted feature maps, emphasizing the most relevant channels. The flattened spatial features are concatenated with non-spatial tokens and passed through a series of linear layers and attention blocks, ultimately producing output.
- **Network3**: Building upon the structure of Network2, this model incorporates Feature-wise Linear Modulation (FiLM) layer [5] to enhance the interaction between spatial and non-spatial information by applying scaling and linear shift to

convolutional layer output. This approach helps in dynamic modulation of input features, filtering irrelevant actions, and contextualizing features, three aspects that are prominent in environment with such a large action space, like Blood Bowl.

B. *Training Performance*

Over the course of training, we tracked validation and training accuracies (Fig. 3).

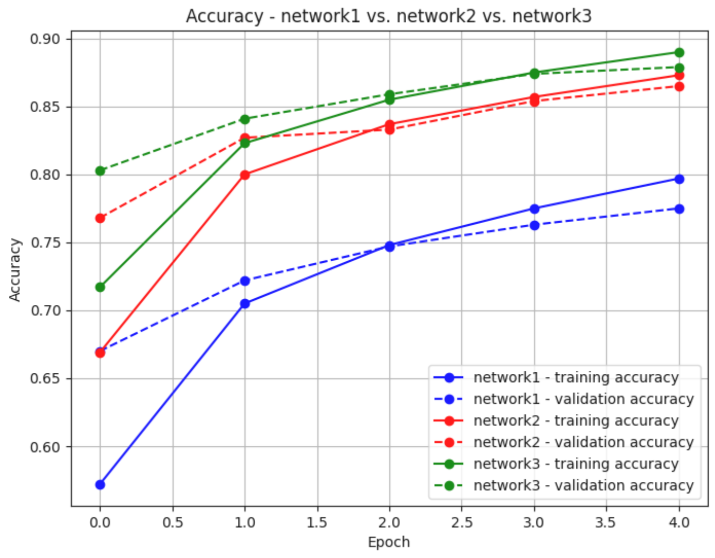

Fig. 3. Accuracies for training and validation.

C. *Match Results*

For competition and data generation, four [1] types of bots were employed:

- A random bot that produces completely random moves.
- A scripted bot in which each action is predetermined.
- The ujibot [6], which generates random sequences of actions, simulates their outcomes in a cloned game environment, and selects the combination yielding the best heuristic score before executing the moves.
- Drefsante bot [7], which analyzes the probability of success of each possible action and uses pathing algorithms to choose the optimal move. Winner of Bot Bowl V - the fifth AI competition in Blood Bowl.

The following table I summarizes the head-to-head matches among the trained networks, a standard scripted bot, a random bot and an ujibot. Each cell reflects the results of 10 matches: wins/draws/losses.

[1]Drefsante Bot due to its Java implementation, has limited compatibility with the original environment, and it is not yet compatible with our tournament settings.

vs	**network1**	**network2**	**network3**	**scripted**	**random**	**ujibot**
network1	–	0/6/4	1/3/6	0/0/10	9/1/0	1/4/5
network2	4/6/0	–	3/2/5	1/2/7	10/0/0	0/4/6
network3	6/3/1	5/2/3	–	0/1/9	10/0/0	1/3/6
scripted	10/0/0	7/2/1	9/1/0	–	10/0/0	6/2/2
random	0/1/9	0/0/10	0/0/10	0/0/10	–	0/0/10
ujibot	5/4/1	6/4/0	6/3/1	2/2/6	10/0/0	–

TABLE I
HEAD-TO-HEAD RESULTS BETWEEN BOTS (FORMAT: WINS / DRAWS / LOSSES)

A broader tournament summary is provided in Table II.

Bot	**Wins**	**Draws**	**Losses**	**Avg TD per game**
network1	11	16	23	0.36
network2	15	17	18	0.72
network3	21	11	18	0.92
scripted	40	6	4	1.78
random	0	1	49	0.00
ujibot	30	15	5	1.54

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT BOTS

VI. IMPACT OF TRAINING DATA

Tests using only scripted self-play data revealed that the bot struggles to secure wins even against no resistance, while data generated by scripted against the random bot failed to compete with a advanced opponents (e.g. it does not know how to intercept the ball).

Several dataset configurations were tested. For example (Datasets 1-3 consist of data dumped from ujibot and scripted and Dataset 4 only from scripted):

- **Dataset 1:** 500 games scripted vs. scripted, 500 games ujibot vs. ujibot, 500 games random vs. scripted.
- **Dataset 2:** 750 games scripted vs. scripted, 750 games ujibot vs. ujibot.
- **Dataset 3:** 750 games ujibot vs. ujibot, 250 games random vs. scripted.
- **Dataset 4:** 250 games random vs. scripted, 250 games scripted vs. random, 500 games scripted vs. scripted, 250 games ujibot vs. scripted, and 250 games scripted vs. ujibot.
- **Dataset 4-mix:** Dataset 4 mixed with 500 games drefsante vs. drefsante.
- **Dataset 4-reload:** In this variant model was first trained on Dataset 4, than reloaded and trained additionally with 500 games drefsante vs. drefsante.

A tournament was conducted with bots trained using the Network3 architecture. Tournament summary is presented in Table III.

Bot	**Wins**	**Draws**	**Losses**	**Avg TD per game**
dataset1	21	27	32	0.60
dataset2	28	30	22	0.74
dataset3	14	22	44	0.29
dataset4	32	23	25	0.90
dataset4-mix	29	30	21	0.84
dataset4-reload	37	26	17	0.95
scripted	50	21	9	1.48
random	0	1	79	0.00
ujibot	46	26	8	1.29

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT BOTS

The head-to-head results are presented in Table IV.

vs	Dataset1	Dataset2	Dataset3	Dataset4	Dataset4-mix	Dataset4-reload	scripted	random	ujibot
Dataset1	–	2/5/3	3/5/2	2/3/5	2/2/6	1/6/3	0/3/7	10/0/0	1/4/5
Dataset2	3/5/2	–	3/6/1	5/2/3	3/5/2	3/3/4	1/3/6	9/1/0	1/5/4
Dataset3	2/5/3	1/6/3	–	0/4/6	1/1/8	0/1/9	0/3/7	10/0/0	0/1/9
Dataset4	5/3/2	3/2/5	6/4/0	–	2/7/1	2/3/5	2/2/6	10/0/0	2/2/6
Dataset4-mix	6/2/2	2/5/3	8/1/1	1/7/2	–	1/6/3	0/4/6	10/0/0	1/5/4
Dataset4-reload	3/6/1	4/3/3	9/1/0	5/3/2	3/6/1	–	3/2/5	10/0/0	0/5/5
scripted	7/3/0	6/3/1	7/3/0	6/2/2	6/4/0	5/2/3	–	10/0/0	3/4/3
random	0/0/10	0/1/9	0/0/10	0/0/10	0/0/10	0/0/10	0/0/10	–	0/0/10
ujibot	5/4/1	4/5/1	9/1/0	6/2/2	4/5/1	5/5/0	3/4/3	10/0/0	–

TABLE IV
HEAD-TO-HEAD RESULTS BETWEEN DATASET-TRAINED BOTS (FORMAT: WINS/DRAWS/LOSSES).

VII. DISCUSSION OF RESULTS

The experiments revealed clear differences in both network architectures and training datasets:

- **Network Architectures:** The architecture inspired by AlphaStar (Network1), which incorporated a Transformer module, showed weaker overall performance than the approach inspired by the MimicBot [4] (Network2). Incorporating FiLM layers (Network3) produced slight improvements in both accuracy and win rate compared to the standard MimicBot-like model, suggesting that feature-wise conditioning can help the network better integrate spatial and non-spatial inputs.
- **Training Data Composition:** Models trained without demonstrations from a weaker opponent (e.g., Dataset2) showed poorer performance, indicating that facing only stronger or similarly strong bots does not provide sufficient state-action diversity. Likewise, training solely on self-play did not yield robust policies capable of handling various play styles. The best results were obtained from Dataset4, which contained demonstrations of the best bot playing against a diverse set of opponents. This variety translated into a higher win rate and more consistent in-game decisions. Additionally, the homogeneity of training strategies can be critical, as demonstrated by Dataset3, where data generated by two different bots, predominantly incorporating random elements, resulted in a lower win rate.
- **Mixing datasets versus reloading approach:** The comparison between mixed dataset and reloaded dataset reveals nuanced insights into model performance. While incorporating a variety of opponents and play styles by mixing, generally help improve generalization and consistency, the reloaded dataset approach actually led to superior performance in the competition. This suggests that retraining the model with progressively more challenging data or focusing on specific opponents can help the model better fine-tune its strategies and adapt to complex scenarios encountered in competition. The success of the reloaded approach emphasizes the importance of targeted retraining to handle more difficult situations, highlighting that sometimes focusing on a narrower, but more challenging, set of data can outperform broad, mixed training. In conclusion, while both strategies have their merits, the reloaded dataset proved more effective in complex environment like BotBowl is, where the model benefits from refined, focused learning on increasingly difficult tasks.
- **Fine-Tuning with Reinforcement Learning:** It is important to note that fine-tuning using reinforcement learning did not yield significant improvements that enhanced the quality of the bots.

In summary, although Behavioral Cloning successfully captures basic strategic patterns, its efficacy heavily depends on the chosen architecture and the diversity of the training data. The FiLM-enhanced model performed better than the Transformer-based one, and using a wide variety of opponents produced the strongest agent. However, the overall performance remains inherently limited by the quality of the training data, and the addition of reinforcement learning fine-tuning did not provide the expected benefits.

REFERENCES

[1] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," CoRR, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[2] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement Learning – Overview of recent progress and implications for process control," *Computers & Chemical Engineering*, vol. 127, pp. 282–294, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135419300754

[3] N. Justesen *et al.*, "Blood Bowl: A New Board Game Challenge and Competition for AI," in *Proc. 2019 IEEE Conf. Games (CoG)*, 2019, pp. 1–8, doi: 10.1109/CIG.2019.8848063.

[4] N. Pezzotti, "MimicBot: Combining Imitation and Reinforcement Learning to win in Bot Bowl," 2021, doi: 10.48550/arXiv.2108.09478. [Online]. Available: https://doi.org/10.48550/arXiv.2108.09478

[5] E. Perez *et al.*, "FiLM: Visual Reasoning with a General Conditioning Layer," arXiv:1709.07871, 2017. [Online]. Available: https://arxiv.org/abs/1709.07871

[6] AdriMon27, "BotBowlUJI," GitHub. [Online]. Available: https://github.com/AdriMon27/BotBowlUJI, accessed: Apr. 2, 2025.

[7] F. Bair, "Drefsante Bot," [Online]. Available: https://drefsante.blogspot.com, accessed: Apr. 2, 2025.

[8] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019, doi: 10.1038/s41586-019-1724-z.

[9] J. Humeau *et al.*, "Planning in the midst of chaos: how a stochastic Blood Bowl model can help to identify key planning features," in *Proc. 2021 IEEE Conf. Games (CoG)*, 2021, pp. 1–4, doi: 10.1109/CoG52621.2021.9619129.