Blood Bowl: A New Board Game Challenge and Competition for AI

Niels Justesen
IT University of Copenhagen
Copenhagen, Denmark
noju@itu.dk

Lasse Møller Uth
IT University of Copenhagen
Copenhagen, Denmark
laut@itu.dk

Christopher Jakobsen
IT University of Copenhagen
Copenhagen, Denmark
chjak@itu.dk

Peter David Moore
Sydney, Australia
peter.d.moore@gmail.com

Julian Togelius
New York University
New York, USA
julian@togelius.com

Sebastian Risi
IT University of Copenhagen
Copenhagen, Denmark
sebr@itu.dk

***Abstract*—We propose the popular board game Blood Bowl as a new challenge for Artificial Intelligence (AI). Blood Bowl is a fully-observable, stochastic, turn-based, modern-style board game with a grid-based game board. At first sight, the game ought to be approachable by numerous game-playing algorithms. However, as all pieces on the board belonging to a player can be moved several times each turn, the *turn-wise* branching factor becomes overwhelming for traditional algorithms. Additionally, scoring points in the game is rare and difficult, which makes it hard to design heuristics for search algorithms or apply reinforcement learning. We present the Fantasy Football AI (FFAI) framework that implements the core rules of Blood Bowl and includes a forward model, several OpenAI Gym environments for reinforcement learning, competition functionalities, and a web application that allows for human play. We also present Bot Bowl I, the first AI competition that will use FFAI along with baseline agents and preliminary reinforcement learning results. Additionally, we present a wealth of opportunities for future AI competitions based on FFAI.**

I. Introduction

Games have proven to be important testbeds for Artificial Intelligence (AI). In the last few years, deep reinforcement learning has enabled computers to learn how to play games such as Chess [14], Go [14], Atari games [11], and many other games [9]. This advance has unfortunately led to a common misconception that computers can now play all interesting board games. In this paper, we propose the popular board game Blood Bowl (Games Workshop, 1986) as the next grand board game challenge for AI. The *turn-wise* branching factor of Blood Bowl is several orders of magnitude larger than those of classic board games, and our experiments have shown that a random agent was unable to score any points in 350,000 Blood Bowl matches, making it infeasible to apply vanilla reinforcement learning. In retrospect, recent AI game challenges such as Go and most Atari games are in fact particularly suitable for deep reinforcement algorithms as they have image, or image-like, observations as well as a fixed action space. Blood Bowl does not have these properties, as observations consist of both spatial and non-spatial information and the available actions depend on the game state, similarly to the StarCraft II Learning Environment (SC2LE) [19]. This paper additionally presents the Fantasy Football AI (FFAI) framework, a Python implementation of Blood Bowl with an API for scripted bots and several OpenAI Gym environments, including scaled down versions of Blood Bowl. We also describe a scripted Blood Bowl bot called GrodBot and present preliminary results of training a deep reinforcement learning agent in three smaller variants of Blood Bowl. Additionally, this paper details plans for several future AI competitions using FFAI. This paper is a significantly extended version of a short paper that initial proposed Blood Bowl as an AI challenge [7].

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

II. Blood Bowl

A. Game Overview

Blood Bowl is a board game designed by Jervis Johnson in 1986 and published by Games Workshop. It is a so-called *fantasy football* game (not to be confused with the American football manager games) that is played on a board of 26×15 squares mimicking a football / rugby-like pitch (Figure 1). Two players each control a team of miniatures and the goal is to score the most touchdowns. We will refer to players as *coaches* (or sometimes *teams* in a traditional sports-like manner) and the miniatures on the board as *players*. Each coach can field 11 players on the board whereafter coaches take turns to move all their players. Players can either move, pass, hand-off, block (attempt to knock down opposing players), blitz (move and block) or foul (stomp on down players) during their *player turn*. When the ball carrier reaches the opponent's end zone their team/coach scores a point.

B. Game Rules

The rules of Blood Bowl have gone through major alterations since the first release in 1986, especially in the 2nd edition (1988), and 3rd edition (1994), whereafter the rules were periodically updated by the Blood Bowl Rules Committee (2002-2009) resulting in the Living Rulebook 6 (LRB6) [4]. The 2016 Edition of Blood Bowl came with a new

ruleset very similar to LRB6. This paper will at all times refer to the rules in the LRB6 as they have been distributed online for free by Games Workshop and are thus easy to obtain.

An important concept in Blood Bowl that plays a role in most aspects of the game is the *tackle zone*. A player's tackle zone consists of the eight surrounding squares in which it is risky for opponent players to perform actions. Examples of the effects of tackle zones are shown in Figure 2. Many coaches make use of the famous *Cage* formation, wherein the ball carrier is surrounded by team-mates, usually positioned on the diagonally adjacent squares, such that any opponent trying to block the ball carrier has to make a hard Dodge roll. Players that are prone, hypnotized, etc. do not have a tackle zone. Another important concept is that players can get *knocked down* either as a result of a block or a failed move near opponent players. When a player is knocked down, an armor roll is typically required: two 6-sided dice (D6) are rolled, and if the result is higher than the player's *Armor Value* (AV) attribute its armor is broken. If this occurs, an injury roll is made to determine if the player is *stunned* (becomes prone and inactive for one turn), *knocked out* of the game (may return at next kickoff), or injured/dead (will not return).

1) Turns: A game of Blood Bowl is separated into two halves, each with eight turns for each coach alternating back and forth. The game starts with a kick-off, where each coach sets up a maximum of eleven players on their respective halves of the pitch and the kicking team places the ball on the receiving team's half. Hereafter, the receiving team's turn starts. Within a turn, each player on the team, that is not *stunned*, can take one of six player actions: Move, Pass, Hand-off, Block (attempt to knock down opposing players), Blitz (move and block), or Foul (stomp on prone players). If a player at any point stands in the opponent's endzone with the ball, it is a touchdown; the scoring team scores one point and both teams must set up for kick-off again. When there are no more turns in the first half, each team similarly sets up again for kick-off. When the second half is over, the team with the most touchdowns wins, or if it is a tie, the game ends in a draw. A critical rule in Blood Bowl is the turnover rule. If any player fails a dice roll, such that the acting player falls over, fails to pick up, or catch the ball, the coach's turn ends immediately.

2) Setup: Before each kickoff, the kicking/defending team first sets up a maximum of eleven players on their half of the pitch. It often happens that a coach cannot field eleven players because of injuries. Besides the maximum number of players on the pitch, there must be a maximum of two players on each wing and a minimum of three players on the line of scrimmage. An example of a defensive zone formation (the blue team) and an offensive wing formation (the red team) is shown in Figure 1. Notice that the offensive team has one player positioned to get the kicked ball in the backfield. Coaches usually have their own repertoire of formations that are tailored to their strategy and playing style.

3) Movement and Dodging: Moving players into positions that give tactical advantage is perhaps the most important part of Blood Bowl. Unless a player is stunned or takes a block

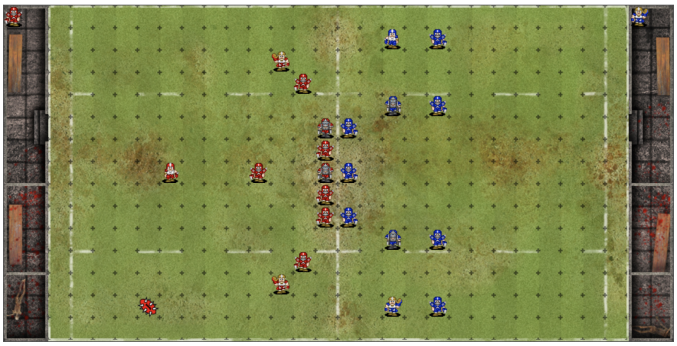

Fig. 1: The game board in FFAI after both teams have set up. The blue team just kicked the ball to the red team and assumed a defensive cover formation, while the red team is in an offensive wedge formation, protecting the wings against blitzing opponents.

(a) (b) (c)

Fig. 2: Effects of tackle zones on different dice rolls visualized in FFAI. (a) The red Lineman can block one of two blue Linemen. When blocking the top-most blue Lineman, two block dice are used due to the two assisting red Blitzers in the top, while it only gets one block die when blocking the bottom-most Lineman. (b) The red Catcher can move to seven adjacent free squares. Since it is already in a blue player's tackle zone, a Dodge roll is required. The player has $AG = 3$, which makes the dodge successful on a roll of 3+. However, this number is increased by one for each opponent tackle zone covering the target square. (c) A red human Thrower can attempt to pass the ball to four nearby team-mates. Two of these team-mates are in the *quick pass* range where a pass will be accurate on a roll of 3+ while the other two are in the *short pass* range requiring a roll of 4+. If the pass is accurate, the ball can be caught on 3+ with an additional modifier for each opponent tackle zone covering the catching player. Note that (b) and (c) show the required dice rolls after modifiers have been added.

action, it is allowed to move a number of squares equal to its *Movement Allowance* (MA) attribute. For example, a Halfling can only move five squares while a Wood Elf Catcher can move eight squares. To move from a square that is within an opponent tackle zone, a player has to pass a Dodge roll that depends on its *Agility* (AG) attribute (thus also called an Agility roll). The higher the agility, the higher the chance is for the roll to succeed. Moving from an opponent tackle zone into other opponent tackle zones add further modifiers to the roll. An example of a tough dodging situation for a red Catcher is shown in Figure 2b. Also, notice in Figure 1 how the defending blue team has positioned itself such that the red team cannot easily run through its cover defense. When a player has moved the number of squares equal to its MA, it has the option to make up to two risky *Going For It* (GFI) moves, allowing the player to move an additional square on a roll of 2 or more. Players that fail a dodge or GFI roll are knocked down. If a player moves to a square with a ball an Agility roll must be made to attempt to pick it up.

4) Blocking, Blitzing and Fouling: Another important part of Blood Bowl is blocking. In fact, some coaches focus more

on blocking, attempting to knock out opponent players, than worrying about the ball. A player taking the Block action can perform a block on an adjacent standing opponent. If the two players have the same value in their *Strength* (ST) attribute, one block die is rolled. If one player is stronger than the other, two (and sometimes three) block dice are rolled, whereafter the stronger player's coach must choose one of the rolled dice to take effect. A block die has six sides with the following outcomes: 1) Attacker Down: the attacking player is knocked down, 2) Both Down: both players are knocked down unless they have the Block skill, 3-4) Push: a player is pushed one square back, 5) Defender Stumbles: the defender is pushed and knocked down unless it has the Dodge skill. 6) Defender Down: the defender is pushed and knocked down. A player on the blocking team can assist the block if it inside blocked player's tackle zone but not in any other tackle zones of the blocked player's team. Likewise, an opponent player can assist the blocked player if it is within the blocking player's tackle zone but not in any other tackle zone of the blocking player's team. Each assist on each side adds one to the attacker or defenders ST. Figure 2a shows an example where a red lineman can get two assists by blocking the top-most blue lineman while the opponent gets a single assist, resulting in two block dice. One Blitz action can be taken each turn, which corresponds to a Move action wherein the player can perform one Block action. Similarly, one Foul action can be made each turn which consists of a Move action followed by a foul. A foul (kicking a player that is down) can only be done to prone players and do not require a block roll. Instead, an armor roll and eventually an injury roll is made directly where assists are applied to modify the armor roll. However, kicking players that are down is strictly against the rules, and thus if either the armor roll or the injury roll is doubles the referee spots the foul and sends the fouling player out of the game.

5) Passing and Hand-offs: A player taking the Pass action can first move as if it was a Move action and thereafter pass the ball. A pass consists of three parts. First, the passer declares a target square. Then the opponent coach declares a player that stands between the passer and the target (if any) who will attempt to intercept the ball. Interceptions are hard and require an Agility roll with additional modifiers. If the interception is successful, the intercepting player catches the ball and its a turnover. If not, the passer must make an Agility roll that depends on the range to the target and the number of opponent tackle zones the passer is in. If that fails, the ball is either fumbled to a nearby square or becomes inaccurate and is scattered three random squares from the target. Otherwise, the ball lands on the target square. If there is a player on the square the ball lands on, that player must finally attempt to catch it by passing an Agility roll. If the pass does not result in a catch by a team-mate it is a turnover. An example of a pass situation is shown in Figure 2c where both the Agility roll for passing and catching is shown for each friendly player. A Hand-off action is similar to a Pass action with a few differences. The ball can only be "passed" to an adjacent square with a team-mate, the "pass" is always accurate, and it cannot be intercepted. A maximum of one pass and one hand-off actions can be performed each turn.

Qty	Title	MA	ST	AG	AV	Skills
0-16	Linemen	6	3	3	8	
0-4	Catchers	8	2	3	7	Dodge, Catch
0-2	Throwers	6	3	3	8	Sure Hands, Pass
0-4	Blitzers	7	3	3	8	Block
0-1	Ogre	6	5	2	9	Loner, Bone-head, Mighty Blow Thick Skull, Throw Team-Mate

TABLE I: The positional players allowed on a Human team. MA=Movement Allowance, ST=Strength, AG=Agility, AV=Armour value. The Ogre is a special type of player called *Big Guys* and its skills are not explained in this paper.

6) Teams and Races: The board game Blood Bowl comes with two teams: the Humans and the Orcs. The Human team does not have any particular strengths nor weaknesses while the Orcs are stronger, have more armor, are slower, and less agile. Each team can be built from a restricted set of positional players (different types of players) with different attributes and skills; any Human team can have up to 16 Linemen, 4 Catchers, 2 Throwers, 4 Blitzers, and 1 Ogre. Table I shows each positional player's attributes and skills. There are 24 teams in LRB6, all with their own strengths and weaknesses. In competitive play, teams are built starting with a *treasury* of 1 million or 1.1 million gold coins, whereafter the roster is created by buying from the available positional players.

C. Competitive Play

Blood Bowl is very popular among competitive tabletop gamers with 161,080 recorded tabletop tournament matches registered by NAF[1], the international association of Blood Bowl coaches and organizers of the Blood Bowl world cup. NAF also maintains the ranking of competitive Blood Bowl coaches based on their participation in tournaments[2]. Not all teams are balanced to be equally good. For example, the Undead team has the record low loss rate of 31.9% and the Ogres team has the highest loss rate of 58.6%.

Blood Bowl is also played competitively in leagues where players (the miniatures) are rewarded experience points for completing passes, scoring touchdowns, and causing casualties. Players can then level up when enough experience points are reached, resulting in new skills or attribute increases. Players can also gain permanent injuries that reduce their value. The coach can in-between matches decide to fire old players and hire new players. A team thus progresses throughout a league, making each league match a unique experience.

A video game adaptation with 3D graphics was released by Cyanide Studios in 2009 which features online play. The video game includes an AI which is far from human-level; it presumably follows a set of scripted rules combined with a pathfinding algorithm. FUMBBL (the acronym combines the football term Fumble with BBL; Blood Bowl League) is a community-driven online league with more than 2,500,000 recorded matches[3]. Matches in FUMBBL are played using an

[1]http://naf.talkfantasyfootball.org/total_for_all_competitions.html
[2]https://www.thenaf.net/rankings/glicko-rankings/
[3]https://fumbbl.com/p/stats

unofficial game client with simple 2D graphics[4]. The source code is kept secret by the developers to prevent cheating.

D. Game Variants

Blood Bowl has several variants using extensions of the core rules[5]. Dungeon Bowl (Games Workshop, 1988) is, as the name suggests, Blood Bowl in a dungeon. There are no fixed rules on the structure of this dungeon, only that it is grid-based, has setup zones and endzones for each team. The rules also include trapped chests, teleporters, lava pits, monsters, etc. that makes every game unique. Other variants exist, such as Death Bowl that allows four teams to play on a cross-shaped pitch with two balls. Street Bowl, Blood Bowl Sevens, Blitz Bowl are smaller variants of Blood Bowl, with a smaller game board and fewer players.

E. Characteristics

This section contains a short analysis of Blood Bowl using the game characteristic dimensions defined by Yannakakis and Togelius [20]. These dimensions include observability (perfect or non-perfect information), stochasticity (deterministic or stochastic), and time granularity (turn-based or real-time). Additionally, we consider at the state representation.

Blood Bowl has **perfect information** as the board state is fully observable and coaches have no hidden information. The game has an optional rule that allows coaches to have secret *special play* cards but these are rarely used in competitions.

Blood Bowl is **stochastic** as most of the interesting actions require dice rolls to succeed. Coaches have a limited number of re-roll tokens they can use to re-roll a failed roll. Experienced Blood Bowl coaches usually start their turn with safe actions that require easy or no dice rolls and postpone risky actions to the end of their turn.

Blood Bowl is a **turn-based** and a **multi-action** game as coaches take turns to move *multiple* players on the board. Other multi-action games that have been the basis for research on AI methods are Arimaa [16] and Hero Academy (Robot Entertainment, 2012) [6]. What makes Blood Bowl even more complicated than these is that players can be moved several steps each turn, making it a *nested* multi-action game. A turn consists of multiple player actions which consists of multiple individual actions. Blood Bowl can be played with a maximum time limit for each turn, usually of two to four minutes per turn, and is sometimes enforced in competitive play.

The **state representation** is especially relevant for deep learning methods. Go and most Atari games are particularly suitable for deep learning methods as they have an image, or image-like, state representation as well as a fixed action space, which is not the case for Blood Bowl. Here, the state is represented by players on the board, each with multiple dimensions (player attributes, whether it is standing, knocked down etc.), a dugout for both coaches with reserve, knocked out, and injured players, weather conditions, and occasionally information on a dice roll, etc. The state representation thus consists of both multi-layered spatial features and non-spatial features very similar to SC2LE [19]. Another similarity shared with SC2LE is that the **action space** varies between steps.

F. Complexity

To get a grasp of the complexity of Blood Bowl we will first attempt to estimate the branching factor analytically and then empirically measure it relative to the game-play of two baseline agents in FFAI.

The size of the action space in Blood Bowl depends on the state and varies between 1 and 395. Sometimes the coach has to select between a few dice results and other times one of the 395 squares on the board to kick or pass the ball to. In most situations, the coach has to select one of up to eight adjacent squares to move a player to or select one of six different action types for a player. A reasonable estimation of the average *step-wise* branching factor is 10 and the average number of steps in a complete player action could be around 5. The number of unique action sequences for just one player action is thus 10^5. With 10 players able to take actions in a fixed order, the average *turn-wise* branching factor is $10^{5\times10} = 10^{50}$. As players can take actions any order, this is a lower-bound estimate. In comparison, the *turn-wise* branching factor is around 30 in Chess and 300 in Go. Long action sequences with sparse scores make both search algorithms and reinforcement learning harder to apply. A game of Blood Bowl consists of approximately $10 \times 5 \times 32 = 1600$ steps, using the previous estimations multiplied by 32 turns.

We performed an experiment in FFAI, simulating 100 games with two agents that samples actions uniformly. 169.8 actions were taken on average by each agent (10.6 actions per turn) with a *step-wise* branching factor of 29.8. This gives us a *turn-wise* branching factor of around 10^{30}. Randomly sampled actions are not representative for human play and we thus also performed an experiment with the scripted bot GrodBot (which will be described in Section IV-B). However, this experiment only includes 10 games because GrodBot runs much slower than the random bot. Here, the measured average *step-wise* branching factor was 17.2 with 41.6 actions per turn and thus the *turn-wise* branching factor is around $17^{42} = 4.8 \times 10^{51}$.

The branching factor was estimated for one setup of Blood Bowl with two simple Human teams while the game has many possible setups; a coach can choose among 24 races to play and has the option to customize the roster. Especially in leagues, it is typical to play against a team with a unique combination of players that the coach has never encountered before, which contrasts sharply with classic board games that have just one or a few initial game states. We describe these different setups in more details in Section VI-B.

G. Motivation

There are several motivations for proposing Blood Bowl as a new AI challenge. First of all, Blood Bowl is, due to its high complexity, significantly harder for AIs to play than classic board games. Having a hard task that is focused on tactical planning, long-term planning, and careful risk

[4]https://fumbbl.com/

[5]https://www.thenaf.net/blood-bowl/variants/

management, without dealing with the array of challenges in real-time video games, is of high value. We can effectively do research on an environment that is fast, easy modifiable and lies somewhere in the large gap between Go and StarCraft in terms of complexity. Exploring video games, which are more similar to real-world problems, are obviously very useful to use as AI test beds. However, by staying in the realm of board games, it is easier to compare the cognitive level of AI systems to humans, which has been argued to be impossible in video games [2]. Another main motivation for proposing Blood Bowl is its expressiveness. Every game of Blood Bowl is never the same. Not only because it quickly branches out to new situations, but because the number of possible initial board states is astronomical, taking into account the number of permutations of players on each of the 24 teams and the relatively non-restricted freedom to set up the players on the board. Additionally, Dungeon bowl introduces the possibility of playing in new and unseen dungeons. These challenges are not in classic board games.

III. Game Engine

Existing Blood Bowl implementations are closed source and do not have an AI interface. Thus, we have developed our own game engine named the *Fantasy Football AI* (FFAI) client[6]. Figure 1 shows a screenshot of a part of the user interface in FFAI with our own 2D graphics[7]. FFAI is implemented in Python allowing a simple way to interface with popular machine learning libraries. We considered implementing the engine in C++ with a Python interface on top, but the state updates in Blood Bowl are fairly simple, and thus fast, even in Python. FFAI implements the Open AI Gym interface for reinforcement learning algorithms [1]. The observation object includes several spatial feature layers as well as several non-spatial features. Aside from the reinforcement learning interface, the engine itself can be used as a forward model. One game step with a randomly sampled action takes on average 0.9ms on a regular laptop and a complete game takes on average 0.16s. While the forward model is rather fast itself, the game state object is object-oriented and thus slow to clone. Tree search is possible, but not very feasible using the current data structure. A simpler array-based data structure could be implemented for this paper while it would complicate the game logic. FFAI also comes with a simple web application that implements a user interface for humans to play against other humans, online or local, or against bots.

A. Competition Functionalities

To support AI competitions, FFAI comes with built-in functionalities to handle a sequence of games between two bots, restricting and penalizing hanging or crashing bots. FFAI can be configured with time limits for a complete turn, a single action taking place in the opponent's turn (such as block die selection or skill usage - we refer to this as an *opponent procedure*), initialization, and termination procedures. The game waits for the acting agent to return an action whereafter it can be penalized in several ways:

- **Delay of Game:** the acting agent fails to end its turn, or returns an action during an opponent procedure, within the time limit. If a Delay of Game penalty occurs, actions are randomly taken by the system until the turn or opponent procedure ends.
- **No Response:** if there is a Delay of Game and the agent responded later than a specified disqualification threshold, it will be disqualified directly.
- **Crash:** if the acting agent crashed, it will be disqualified directly. In the unexpected case that FFAI crashed during an internal procedure, the game will end in draw.

FFAI also saves a report of the competition with aggregated results and a list of individual game results.

B. Replays

We are currently working on a module for FFAI to replay matches from FUMBBL. This would enable us to extract state and action pairs from the 2,500,000 available FUMBBL matches which can be used for imitation learning. Here, the goal is to learn a policy function that maps states to actions using traditional supervised learning techniques, thus imitating a the playing styles expressed in the dataset. This technique has been applied to several games, including StarCraft [5], and Candy Crush Saga [3]. Currently, individual replays can be fetched from the FUMBBL API. However, this process is slow and we are thus planning to release publicly available data sets in the future.

IV. Baseline Agents

A. Random Agent

FFAI comes with an agent that samples actions from a uniform distribution. To be more precise, it first samples a legal action type uniformly and if that action type requires a position as well, such as a movement action, a legal position is sampled uniformly as well. When setting up, it randomly selects between two predefined formations on the offense and two on the defense. In 350,000 games against itself, no touchdowns were scored and thus all games ended in a draw. Games in which random agents practically never score points or wins are extremely challenging for many algorithms such as Monte Carlo Tree Search and Q-learning as they rely on random exploration. Thus, we do not expect vanilla implementations of such algorithms to score any points either.

B. GrodBot

GrodBot[8] is a scripted bot with an estimated skill level slightly higher than a rookie player. GrodBot repeatedly evaluates all possible moves for all players. The move with the highest score is then executed at each step in the game. The *end turn* action is always assigned a score of zero so that the turn will be passed when no positive actions are left. The scoring of moves first applies pathfinding to identify

[6]https://github.com/njustesen/ffai

[7]Nicholas Kelsch has the copyright to the player icons.

[8]https://github.com/njustesen/ffai/blob/master/examples/grodbot.py

the set of possible squares a player can reach along with a probability of success for moving there. The probability of successfully moving to each reachable square is thus the maximum probability of all possible paths leading to it; we always follow the optimal path. GrodBot maintains a list of possible purposes, their values, and some rules that apply modifiers to the values (e.g. it is preferred to pass the ball to an unused team-mate close to the end-zone rather than a used team-mate in the backfield). The final score for a move is then computed by multiplying the probability of success with the modified value of the move. Each purpose was initially valued using an experienced Blood Bowl player's intuition whereafter it was gradually improved by playing GrodBot against itself and tune the values in an ad-hoc manner. GrodBot's move purposes are:

- **Move** to a receiving position
- **Move** to the ball
- **Move** to a player holding the ball somewhere safe or towards the end zone
- **Move** to a player to form a defensive sweep position
- **Move** to a player to form a defensive screening position
- **Move** to a player to form an offensive screening position
- **Move** to a player to form a cage around the ball
- **Move** to a player to exert an opponent tackle zone
- **Foul** an opponent on the ground
- **Blitz** an opponent (preferably the ball carrier)
- **Hand-off** the ball to a team-mate
- **Pass** the ball to a team-mate
- **Block** an adjacent opponent using two block dice

The probability of a successful move in Blood Bowl depends on a number of different factors, such as the number of tackle zones it moves trough, or whether or not the player needs to attempt a GFI. We do not believe there is a useful admissible heuristic for path finding in Blood Bowl and we therefore apply Dijkstra's algorithm. The cost function is probability-based in the interval [0,1] and is not additive. For sequences of moves, their costs are simply multiplied. In general, if $C(\{s_1, s_2\})$ is the cost of the path going from square s_1 to square s_2 and the following cost of going from s_2 to s_3 is $C(\{s_2, s_3\})$, then the total cost of the path from s_1 to s_3 is $C(\{s_1, s_2, s_3\}) = 1 - (1 - C(\{s_1, s_2\})) \times (1 - C(\{s_2, s_3\}))$. The probabilities of success are multiplied and subtract from 1 to convert them back to the probability of failure, i.e. the cos). GrodBot's path-finding functionality has become a part of FFAI and can easily be used by other bots.

We tested GrodBot in ten games against the random baseline (five as the away team and five as the home team) where both agents controlled the basic human team. Grodbot won all ten games with an average of 4.2 touchdowns and 1 inflicted casualty a game. The random agent scores 0 touchdowns and had 0.6 inflicted casualties per game. The number of inflicted causalities also include casualties inflicted by the crowed, failed dodges, etc.

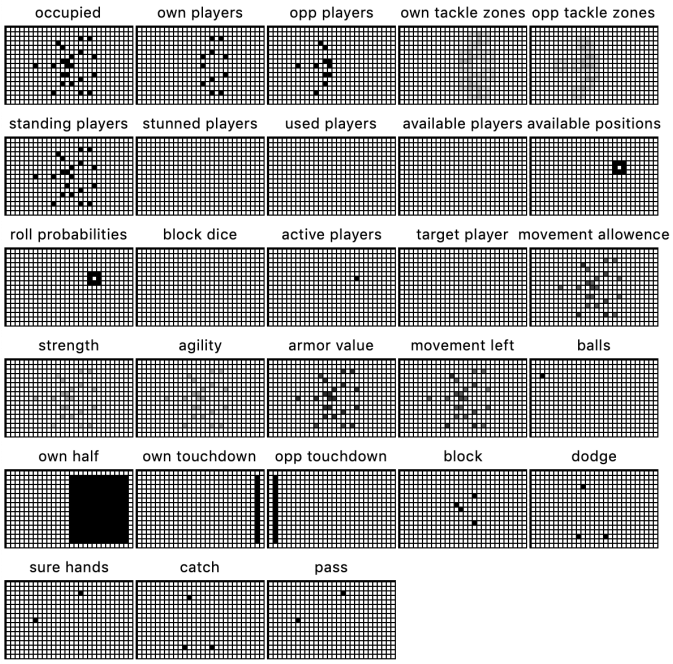

Fig. 3: The 28 spatial feature layers in the FFAI Gym observation. Each layer has a name, which is shown above the visualization. Here, black squares represent a value of 1 and white squares represent a value of 0.

V. FFAI Gym

A. The Gym Interface

OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms by implementing a simple interface to handle communication between the agent and the environment [1]. FFAI comes with several Gym environments, one with the original game board and 11 players on each side and several smaller variants (see Figure 4). In these environments, the agent faces the random agent (see Section IV-A), which probably will lead to sub-optimal behaviors against stronger agents. It is, however, easy to modify the FFAI Gym implementation to e.g. support self-play, which is an effective learning method in two-player board games [15]. Both agents control a basic Human team, but this can also be modified easily. For simplicity, the agent always plays as the home team, playing on the right side of the field. This limitation can easily be resolved after training by simply flipping the board and swapping a few values in the observations, if it has to play as the away team.

The observation space is split into three parts; (1) a vector containing 50 non-spatial normalized values representing the score, turn number, half, re-rolls left, etc. (2) a one-hot encoded vector representing the *procedure* (phase) the game is in, e.g. *Setup*, *Turn*, *PlayerAction*, and *Push*, and (3) a set of 2D feature layers, each with one value per square on the board such as whether a square is occupied by the ball or by a player controlled by the agent or the opponent. All 28 features layers can be seen in Figure 3 for the same game state. The design of the observation space is very similar to that of SC2LE [19] as it also has spatial and non-spatial components.

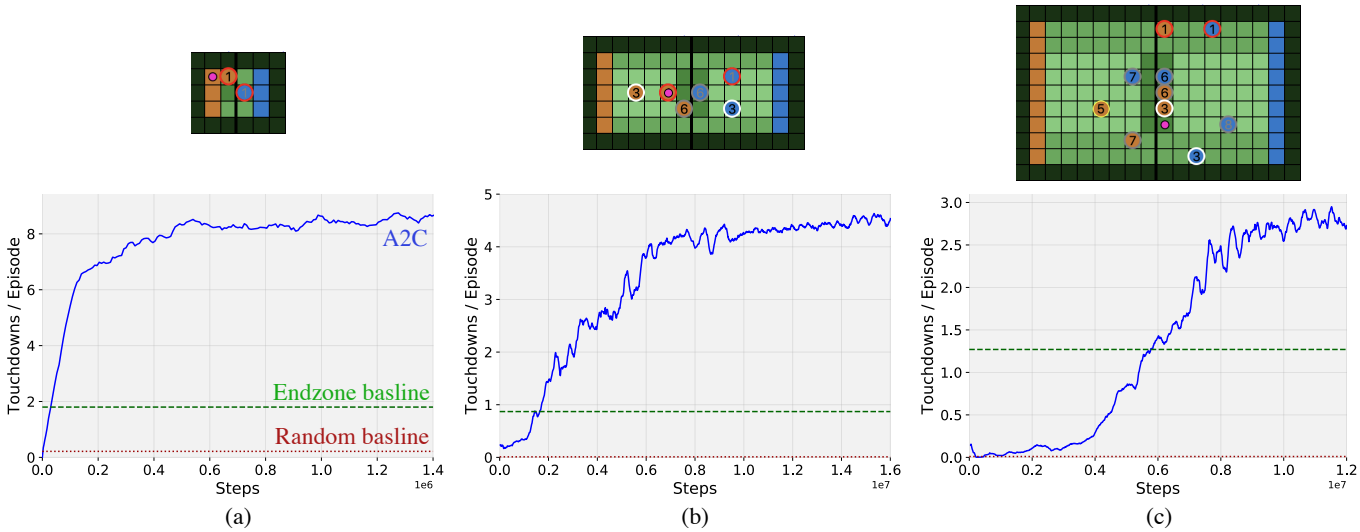

Fig. 4: Touchdowns per episode of A2C during training in the three smallest FFAI Gym environments: (a) FFAI-1-v1, (b) FFAI-1-v3, and (c) FFAI-1-v5, which features 1, 3, and 5 players on the pitch for each team. Simple renderings of each environment is shown above the plots. The agent plays against an agent that takes random actions. The touchdowns are smoothed over 200,000 steps. The red and green lines show the touchdowns per episode the Random and Endzone baselines. We see that A2C learns a policy that is better that the baselines in all three envrionments.

The action space has two parts: (1) the type of the action to perform among 31 choices such as *Block*, *Select Both Down* (block dice result), *Use Reroll*, *Heads* (for the coin toss), etc., and (2) a position, which is only relevant for some action types, e.g. a *Block* action requires a position to determine which opponent player to block.

The built-in reward function only gives a reward of 1 when winning the game, -1 when losing, and 0 otherwise. The complete game state is, however, accessible in the environment to allow for reward shaping.

B. Preliminary Results with A2C

We applied the deep reinforcement learning algorithm synchronous advantage actor-critic [12] (A2C) on the three smallest/easiest Gym environments in FFAI, which has a board of 4×3, 12×5, and 16×9 squares with 1, 3, and 5 fielded players on each team, respectively (see Figure 4). On the two smaller environments, we used only touchdowns as rewards. However, on the larger one, touchdowns are rarely obtained, and thus the following reward shaping was used (rewards are shown in brackets): winning (5), touchdown (4), knock out opponent (3), push opponent into the crowd (3), completion (3), cause opponent fumble (2), knock down opponent (2), handoff (2), opponent failing a dodge (1), moving ball carrier closer to the opponent endzone (1), and gaining the ball (1).

We use a convolutional neural network with an additional fully-connected input stream to handle the non-spatial features. We use all the 28 spatial feature layers from FFAI for the convolutional input stream and all the 50 non-spatial features (in the current version of FFAI there are now 50 non-spatial features) for the fully-connected input stream. The convolutional stream has two layers, one with 16 filters of size 3×3 (mimicking the tackle zone area around a player), and a second layer with 32 filters of size 2×2. Both layers use stride 1 and padding, preserving the spatial structure of the observation. The non-spatial input stream consists of a single fully connected layer of size 25 which is concatenated with the flattened convolutional stream followed by a single fully-connected layer and two output streams for the critic and the policy. When actions are sampled from the policy distribution, we use a masking technique to filter out illegal actions before applying softmax.

A2C was configured to use eight parallel workers, a learning rate of 0.001, a discount factor $\gamma = 0.99$, entropy coefficient of 0.01, value loss coefficient of 0.5, max. gradient normalization of 0.5, and steps per update $t_{max} = 10$. We used the RMSprop optimizer [17] with ϵ = 1e-8 and α = 0.99. Figure 4 shows the touchdowns per episode during training, where it reached an average around 8/5/3 in the three environments. A random baseline agent scores around 0.2/0.0/0.0 touchdowns per episode, and a simple *endzone* baseline that always goes toward the endzone with the ball scores around 1.8/0.9/1.3 touchdowns/episode. The code is made available online[9]. When observing the trained agents play[10] we can see that it plays quite well but not optimal.

VI. AI Competition

A. Bot Bowl I

Based on our analysis of Blood Bowl we believe it can offer a new and exciting testbed for AI due to the high complexity of the game while still resembling many classic board games. To encourage researchers and hobbyists to explore algorithms that can play Blood Bowl, we are planning to organize an annual AI competition with progressively more difficult challenges using FFAI. The first competition, Bot Bowl I[11], will be held

[9]https://github.com/lasseuth1/blood_bowl2/
[10]https://youtu.be/xk4AMutyuaA & https://youtu.be/7xmwB8hn3qM
[11]https://bot-bowl.com/

at the IEEE Conference on Games in August 2019. The competition will allow all types of methods, including controllers that are scripted, search-based, neural network-based, as well as hybrids combining several of these approaches. Bot Bowl I will only allow a pre-fixed Human team, to keep the format of the first competition simple. The submitted agents will play in a round-robin tournament with several matches against each other and the two best agents (with the most wins) will play in a final, also consisting of several matches.

B. Future Competition Formats

Future competitions can be extended in several exciting directions that will add further challenges for the competitors. Four concrete ideas are:

- **All teams:** The most obvious extension of the current competition format is to allow all teams. This addition will require a higher flexibility with less hard-coded strategies and formations.
- **Custom rosters:** To align our competition format with real Blood Bowl tournaments, competitors should be allowed to buy players for their roster from a starting treasury. As agents need to generalize to possibly unseen teams, it will further require agents to rely less on hard-coded strategies.
- **League format:** In Blood Bowl leagues, players get star player points and can gain new skills when leveling up between games. Additionally, players can get permanent injuries. This format thus adds a meta-game, as agents must manage their team in-between games, selecting new skills when players level up, hire new players, etc.
- **Dungeon Bowl:** Instead of playing on the same board, competitions could be played on procedurally generated dungeons using the official Dungeon Bowl rules while building on the numerous algorithms for dungeon generation in the procedural content generation literature [10, 13, 18]. By playing on procedurally generated *unseen* dungeons in the competition, agents need to either use extensive forward search or learn a policy on a vast training set of dungeons in a similar fashion to [8].

These four suggestions for future competitions add new challenges that all require agents to generalize to new game configurations, something that current reinforcement learning algorithms struggle with.

REFERENCES

[1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[2] R. Canaan, C. Salge, J. Togelius, and A. Nealen. Leveling the playing field-fairness in ai versus human game benchmarks. *arXiv preprint arXiv:1903.07008*, 2019.

[3] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao. Human-like playtesting with deep learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[4] J. Johnson. Blood bowl handbook: Blood bowl living rulebook 6.0, 2016.

[5] N. Justesen and S. Risi. Learning macromanagement in starcraft from replays using deep learning. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 162–169. IEEE, 2017.

[6] N. Justesen, T. Mahlmann, S. Risi, and J. Togelius. Playing multi-action adversarial games: Online evolutionary planning versus tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.

[7] N. Justesen, S. Risi, and J. Togelius. Blood bowl: The next board game challenge for ai. In *Foundations of Digital Games*. Association for Computing Machinery, 2018.

[8] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

[9] N. Justesen, P. Bontrager, J. Togelius, and S. Risi. Deep learning for video game playing. *IEEE Transactions on Games*, 2019.

[10] A. Liapis, C. Holmgård, G. N. Yannakakis, and J. Togelius. Procedural personas as critics for dungeon generation. In *European Conference on the Applications of Evolutionary Computation*, pages 331–343. Springer, 2015.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[13] N. Shaker, J. Togelius, and M. J. Nelson. *Procedural content generation in games*. Springer, 2016.

[14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[15] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[16] O. Syed and A. Syed. Arimaa-a new game designed to be difficult for computers. *ICGA Journal*, 26(2):138–139, 2003.

[17] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[18] R. Van Der Linden, R. Lopes, and R. Bidarra. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78–89, 2014.

[19] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. Starcraft ii: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[20] G. N. Yannakakis and J. Togelius. *Artificial Intelligence and Games*. Springer, 2018. http://gameaibook.org.