ACCEPTED MANUSCRIPT

Deep ensemble learning of tactics to control the main force in a real-time strategy game

This Accepted Manuscript (AM) is a PDF file of the manuscript accepted for publication after peer review, when applicable, but does not reflect post-acceptance improvements, or any corrections. Use of this AM is subject to the publisher's embargo period and AM terms of use. Under no circumstances may this AM be shared or distributed under a Creative Commons or other form of open access license, nor may it be reformatted or enhanced, whether by the Author or third parties. By using this AM (for example, by accessing or downloading) you agree to abide by Springer Nature's terms of use for AM versions of subscription articles: https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

The Version of Record (VOR) of this article, as published and maintained by the publisher, is available online at: https://doi.org/10.1007/s11042-023-15742-x. The VOR is the version of the article after copy-editing and typesetting, and connected to open research data, open protocols, and open code where available. Any supplementary information can be found on the journal website, connected to the VOR.

For research integrity purposes it is best practice to cite the published Version of Record (VOR), where available (for example, see ICMJE's guidelines on overlapping publications). Where users do not have access to the VOR, any citation must clearly indicate that the reference is to an Accepted Manuscript (AM) version.

Accepted manuscript

ACCEPTED MANUSCRIPT

Deep Ensemble Learning of Tactics to Control the Main Force in a Real-time Strategy Game

Isaac Han[1] and Kyung-Joong Kim[1*]

[1*]School of Integrated Technology, Gwangju Institute of Science and Technology, 123, Cheomdangwagi-ro, Buk-gu, 61005, Gwangju, South Korea.

*Corresponding author(s). E-mail(s): kjkim@gist.ac.kr; Contributing authors: Issac7778@gm.gist.ac.kr;

Abstract

Professional StarCraft game players are likely to focus on the management of the most important group of units (called the main force) during gameplay. Although macro-level skills have been observed in human game replays, there has been little study of the high-level knowledge used for tactical decision-making, nor exploitation thereof to create AI modules. In this paper, we propose a novel tactical decision-making model that makes decisions to control the main force. We categorized the future movement direction of the main force into six classes (e.g., toward the enemy's main base). The model learned to predict the next destination of the main force based on the large amount of experience represented in replays of human games. To obtain training data, we extracted information from 12,057 replay files produced by human players and obtained the position and movement direction of the main forces through a novel detection algorithm. We applied convolutional neural networks and a Vision Transformer to deal with the high-dimensional state representation and large state spaces. Furthermore, we analyzed human tactics relating to the main force. Model learning success rates of 88.5%, 76.8%, and 56.9% were achieved for the top-3, -2, and -1 accuracies, respectively. The results show that our method is capable of learning human macro-level intentions in real-time strategy games.

Keywords: Decision-making, Deep learning, Real-time strategy game, StarCraft

Accepted manuscript

ACCEPTED MANUSCRIPT

1

Accepted manuscript

ACCEPTED MANUSCRIPT

1 Introduction

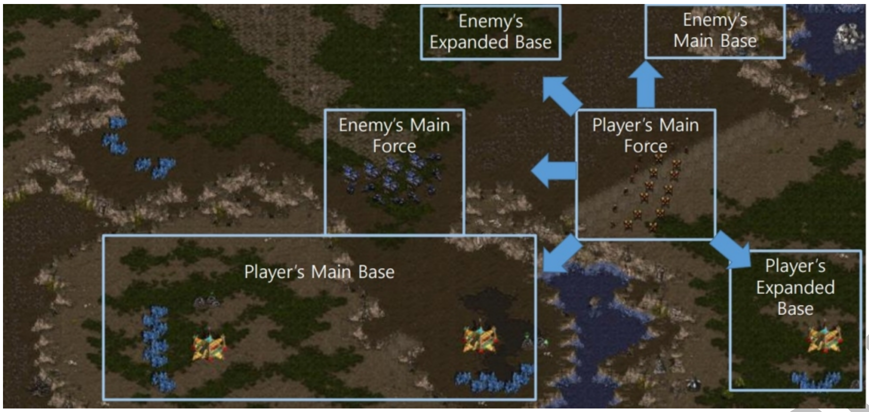

Fig. 1 Illustration of the main base, main force, and expansion base. Tactical decisions include the future direction of movement of the player's main force with respect to six categories (main base, expansion base, and "Stop", as well as the enemy's main force, main base, and expansion base). Typically, any expansion close to the main base is known as a "front yard" and it is considered part of the main base in this study. For illustrative purposes, we do not depict the fog of war, which limits the view to the immediate surroundings of the player's units and buildings.

In the last decade, artificial intelligence (AI) has been extensively studied for real-time strategy (RTS) games, because these games pose various challenging problems owing to their large state and action spaces and real-time actions. In RTS games, players perform several decisions simultaneously, such as managing resources, scouting an opponent's territory, controlling units, constructing buildings, and engaging in combat [1]. The most popular series of RTS games include StarCraft, Warcraft, Company of Heroes, Total War, and Command & Conquer.

In RTS gameplay, players need to enact tactical decisions regarding where, when, and how to attack enemy units and structures under uncertain conditions. As such, devising tactical decisions while managing the enemy's tactics with limited information is an extremely challenging problem [2]. To this end, previous studies have proposed a tactical decision-making method for RTS games, such as a Bayesian approach [3] and a Monte Carlo tree search [4]; however, these methods have yielded only simple heuristics-based models. In contrast, deep learning can manage high-dimensional data, to create complex models that can overcome the weaknesses of such heuristics-based models. Few studies have applied deep learning to predict the tactical decisions in RTS games. In particular, each of these studies considered various kinds of tactical decisions such as predicting where to attack or defend [5], types of units or buildings to attack [6], and selecting one of the handcrafted scripts [7].

This study focused on the tactics related to the main force, i.e., managing units grouped as forces is a common strategy in RTS games. Thus, determining

Accepted manuscript

ACCEPTED MANUSCRIPT

the actions of forces is an essential tactical aspect and controlling the main force is particularly important, because it is the strongest force. We interviewed a professional human player, which revealed that the decision-making processes related to the main force were the most important after the very early stage of the initial build order execution in StarCraft. In addition, the present analysis of human replay data revealed that 40% of the units pertain to the main force and account for 60% of the influence.

The proposed method employed deep learning models to learn the decision-making skills of expert human players from replay files and predict the upcoming movement direction of the main force. We acquired nearly 12,000 replays from online sources and extracted 3 million data tuples. Moreover, we provided a comprehensive analysis of the data, and the results revealed that the main force is essential for the game. To efficiently predict the movement direction, we categorized the destination of the main force into six categories. We trained an ensemble model comprising CNNs, a ViT, and decision tree, and the proposed model outperformed all the existing deep learning-based tactical decision-making methods. To the best of our knowledge, this is the first demonstration of direct machine learning of tactical decisions pertaining to the main force based on replays.

The contributions of this study are stated as follows:

- We propose a novel approach for learning vital human skills from replay files, namely, deciding the upcoming direction of the main force. We developed an ensemble model comprising CNNs, a ViT, and a decision tree using approximately 3 million data tuples. The difficulty of learning was minimized through a new input data representation, including the considerations of match type (Protoss [P] vs. Terran [T], P vs. Zerg [Z], etc.). The experimental results revealed that the proposed method achieved learning success rates of up to 92.9%, 81.4%, and 63.3% for the top-3, -2, and -1 accuracies, respectively.
- We analyzed human decision-making processes related to the main force and developed a novel algorithm for main force detection. In addition, we extracted approximately 3 million data samples from 12,057 replays of human games and constructed a dataset to predict the direct of the main force.

2 Background

2.1 StarCraft AI

StarCraft has been important for RTS game research since 2010. The Brood War Application Programming Interface (BWAPI) allows researchers to implement customizable AI players into StarCraft: Brood War, which is one of the most successful RTS games. BWAPI is not an official interface provided by the developer of StarCraft, Blizzard, but it nevertheless allows for the

Accepted manuscript

ACCEPTED MANUSCRIPT

creation of AI players. Since 2010, the AI community has organized international StarCraft AI competitions and conferences, which have helped promote the development of AI techniques for the RTS game community [8] [9]. RTS researchers have used StarCraft to test their algorithms [10] [11]. The game's popularity has attracted many researchers. Also, it is easy to access replays of human games on gaming portals, and it is possible to extract game states and the actions of human players from replay files using BWAPI. Several platforms are available for RTS research, each with its own unique properties [12] [13] [14] [15]. StarCraft is renowned for its popularity, commercial success, and international AI competitions. Due to its popularity since the 1990s, large volumes of replay files are available, of games involving players ranging from novice to professional. Unlike other RTS platforms, Blizzard developed StarCraft for commercial purposes. Recently, Blizzard and DeepMind released a programming interface for the StarCraft II game environment [16].

To build an AI player for StarCraft, many different skills are required. The player should make high-level decisions, such as strategy or tactics, but also undertake low-level control like micromanagement. Due to this wide range of required skills, StarCraft AI research has been conducted to develop modules that perform well in each task and have integrated them into a complete AI agent. These are the skills mostly studied in the StarCraft AI community: build order, tactics, macromanagement, and micromanagement.

In the early stage of a match, it is important to select a build order (the sequence of player actions for producing buildings and units). Professional players usually optimize their actions in this stage, where small mistakes can make a match difficult to win. Early methods are usually based on heuristic search algorithms [17]. Another line of research is using evolutionary computation, which treats a list of build orders as individuals of a population [18] [19]. Recent works have dealt with more challenging tasks, such as considering diversity [20] or planning the build order in an online manner [21].

Making tactical decisions is determining when, where, and how to attack enemy buildings and units. Early approaches for tactics have proposed simple models based on heuristics [3] [4]. Deep learning has been used in recent works, with RL [6] [5]. Although the main force plays an essential role in tactics, previous approaches have not addressed tactics with regard to the main force. Organizing units into a number of forces is a natural way to control units efficiently for StarCraft players. In addition, professional players often run the main force, which is the strongest one among the forces. They create the main force by putting more units and stronger units in a certain force. According to the analysis in this study, the main force takes about 60% of the influences of the total number of units.

Macromanagement is managing high-level strategies, which may include build order optimization or tactics. Various tasks belonging to macromanagement have been addressed in previous works. The authors of [22] have used deep RL to decide what unit to produce in a certain state. Initial strategy has a big influence in StarCraft. Accordingly, the Bayesian approach has been used

Accepted manuscript

ACCEPTED MANUSCRIPT

to predict the opponent's initial strategy [23]. Also, effects of the fog of war have been studied in terms of the opponent's strategy prediction [24]. Replays have been used to train a model that predicts strategy and build order [25]. The probabilistic approach is pervasive in macromanagement [26] [23], but recent works commonly use deep learning [27].

In combat situations, micro-level management of attack units can make a critical difference to the outcome. Micromanagement, involving rapid manipulation of individual units to win in combat is an essential skill when playing StarCraft. A variety of methods have been applied for micromanagement. One of the approaches used is the search-based method. Many studies have proposed search-based algorithms for micromanagement.

The earlier approach is based on Alpha-Beta search, which is an efficient technique relying upon heuristic-based state evaluation function [28]. The authors have made improvements through transposition tables and iterative deepening. The algorithm applied on RTS game bots and achieved enhanced results [29]. Also they have proposed an improved version of UCT (Upper Confidence Bound) search method and a novel search algorithm that greedly search number of portfolios [30]. However, search-based methods suffer from low speed, making them unusable in real time. Hierarchical Adversarial Search (HAS) has been proposed to solve this issue [31]. HAS deals with large state-action space through a hierarchical search paradigm. Another line of research optimizes policies using evolutionary computation or RL. The policies can be represented as potential fields [32], and neural networks [33] [34]. In an aspect of RL, micromanagement is primarily considered a multi-agent problem, which deems each unit as an agent [35]. According to this, micromanagement problems have been used as an evaluation environment for multi-agent RL [36].

In addition to these approaches for developing intelligent modules for a variety of tasks, a comprehensive method has been proposed to achieve expert-level gameplay in StarCraft. An easy way to develop comprehensive AI is to integrate each module into a single agent [37] [5]. Another way is to train using a single comprehensive policy. Multi-agent RL was used to realize AI with Grandmaster level StarCraft II ability [38]. Moreover, StarCraft is also being used for many application domains, such as web-based interface [39], and an algorithm for automatically spectating games [40].

2.1.1 StarCraft AI using Replays

Replays of human StarCraft games have great potential for overcoming the weaknesses of AI bots. Thousands of replay files for StarCraft are readily available on gaming portals, and they include games involving players with different ability levels. Replay extraction software allows AI researchers to extract all the actions taken by human players from these replay files. Massive amounts of sample data allow machine learning models to imitate the decision-making processes of human players. Many studies have utilized replays of human games to build machine learning models to address various problems in RTS games. For example, RL has been applied to micromanagement, with

Accepted manuscript

ACCEPTED MANUSCRIPT

supervised pre-training performed based on replays of human games [41], and a neural network trained with 2,005 replays from professional human players was used for macromanagement [42]. In macromanagement, more recent works have utilized a transformer [43] and replays [27] to train the models. The Defog-GAN model, which reveals hidden information in StarCraft, was trained with replays of human games [44]. Facebook has also obtained 65,000 replays (360 GB of data; the STARDATA project [45]). From these data, a team working at Facebook extracted the total number of units created, match lengths, ratios of resources mined by each player, and opening clusters. Furthermore the authors of [46] have proposed feature extraction method to predict the outcome of the StarCraft league. Despite replays frequently being used for building AI for StarCraft, direct learning of tactics from replays of human games has not been studied previously.

2.2 Tactical Decision-Making

Making tactical decisions using machine learning models has been widely studied in various domains. The representative one among the domains is RTS games, which require strategic planning and tactics for the gameplay. The authors of [5] proposed a modular architecture that can be trained by reinforcement learning and a fully convolutional network (FCN) as their tactics module. However, it considers where to move all units, not just the main force. In addition, the authors of [6] selected macro action using RL. They set 54 macro actions, with 17 of them related to tactics. But those actions are to attack certain types of enemy units or buildings and are not related to the attack location nor the main force. Also, there has been an approach based on supervised learning, not RL. The authors of [7] proposed a tactical decision-making module supervised by a search algorithm, which selects one of the handcrafted scripts.

On the other hand, there have been many works on computational approaches for tactical decision-making in various domains. For autonomous vehicles, the decisions should be made carefully for safety purposes. Many recent studies in tactical decision-making for autonomous driving address this issue using deep learning along with RL [47] [48]. Tactical decisions also play an essential role in team sports. Few studies have used deep neural networks to support tactical decisions for team sports, such as rugby [49] and football [50]. In contrast, deep learning is relatively less applied to make tactical decisions in StarCaft AI research.

2.3 Deep learning for games

CNNs effectively process image data and are thus widely used in game AI, to build machine learning models for games. Also, a Vision Transformer has recently been used for image processing, and it performs much better than CNNs.

Accepted manuscript

ACCEPTED MANUSCRIPT

Many studies have used CNNs to train AI players for various games. Recently, successful results have been obtained using CNNs as approximating functions for game state–action evaluation. For example, DeepMind used a CNN as a Q value approximation function in its deep Q-Learning algorithm. This algorithm can learn how to play Atari games at human levels of skill [51]. Unlike the traditional AI approach, the algorithm uses the raw game screen as input for the CNN. In 2016, DeepMind trained a CNN to predict the decisions of professional Go players, achieving an accuracy of 57% [52]. The Alpha Go system beat a professional Go player in March 2016, thus demonstrating superior performance of the CNN. CNNs that involve RL have been used to play various games. A CNN that involves asynchronous RL was used to evaluate the racing game known as Torcs [53]. RL with a recurrent convolutional architecture was used to play a first-person shooter game [54]. The well-known Minecraft game is used to train RL agents [55]. The multiplayer online battle arena (MOBA) game is both complex and challenging in terms of AI. MOBA has been played using deep RL [56]. Text-based games are also used to test the performance of novel RL agents [57]. Montezuma's Revenge is used to evaluate the exploratory capacities of RL agents [58]. CNNs are used to build human-like AI agents for automated game testing [59]. Human player datasets have been used to train CNN agents in match-3 gaming. The RL approach has been used for the strategic play of match-3 games [60]. CNNs are used to build an AI agent for the Othello game [61]. The authors of [62] used CNNs as an evaluation function to train an RL agent.

Buro et al. [63] used an RTS as a simplified testbed for AI algorithms. The cited authors used twenty-five 10 × 10 planes ("channels" in our study) as inputs for the CNN; they attempted to predict the winner of a match using match data from 15 AI bots. They also tested the CNN as an evaluation function for search algorithms. Justesen et al. trained a neural network using 2,005 replays and achieved a prediction accuracy of 45.4% with respect to the next build action [42]. CNNs have been used to predict the winners of the RTS game [64].

CNN-based generative models have recently been used to improve automatic game content generation. A generative adversarial network [65] is a representative deep generative model widely used in various fields of AI. A generative adversarial network has been used to generate game content and to automatically create game levels [66]. The authors of [67] used CNNs to procedurally generate such levels, then compared them to manually designed levels. The model successfully generated content. The authors of [68] used an RL-based approach to create procedural content [68]; generation was regarded as a sequential problem. In the present work, a CNN was used as a perception module for the RL agent. The quality of the generated content was optimized by the RL; two-dimensional levels were successfully generated.

CNNs are used to perform various game-related tasks, such as that of predicting human behavior in size-variant repeated games [69], or game difficulty. CNNs can also detect residual glitches in video games [70].

Accepted manuscript

ACCEPTED MANUSCRIPT

"Transformer" [43] was initially developed for natural language processing, but it has found applications in image processing [71] and RL [72]. Transformer has been applied in games like text adventure games [73], Go [74], and Viz-Doom [75]. For StarCraft, Transformer has been used for macromanagement [27] and micromanagement. The authors of [27] used Transformer to predict macro actions. They show that Transformer generalizes well to unseen data, like an unseen match type (Terran vs Protoss). Transformer has also been used with RL for micromanagement in which the method approximates to the transformer-based joint action-value function that shows better performance in micromanagement scenarios. However, ViT is very recent and has found only a few applications in games [72].

Compared with these previous investigations, our study differs in certain respects. Unlike previous works, we consider the main force in tactics because, according to our results, it plays an essential role in this area. We apply state-of-the-art deep learning models like CNN and ViT to handle high- dimensional data.

3 Methods

During gameplay, human players gather information, such as the current combat situation, enemy location, and buildings constructed. They then use this information to assess the situation and make decisions. We conducted interviews with a single professional human player to understand their decision-making strategies. The interviews were semi-structured and consisted of a series of open-ended questions designed to elicit information about the players' decision-making strategies related to the main force. We have found that the main force plays an important role in tactical decision-making in StarCraft, especially after the very early stage of the initial build order execution in StarCraft. Also, we have found that the movement of the main force can be categorized into several directions.

Recently, the ability of CNNs to process high-dimensional data at the level that can be processed by humans has been confirmed in terms of both object recognition and gameplay [51]. Notably, the ViT has shown promise when used for image recognition [71]. Thus, in the present study, we used a CNN and ViT to develop a tactical decision-making model, in which the CNN and Transformer process high-dimensional input, namely information from StarCraft. We present an effective decision-making model that operates in a manner similar to a human player.

3.1 Detecting the Main Force

Attack strength differs according to the positions and numbers of attack units. For example, a group of strong attack units is likely to constitute a powerful army if concentrated in one location. However, it also differs by type of the units, like in case one forces consist of many weak units and another consist of few powerful units. In this case it is ambiguous to determine which is the

Accepted manuscript

ACCEPTED MANUSCRIPT

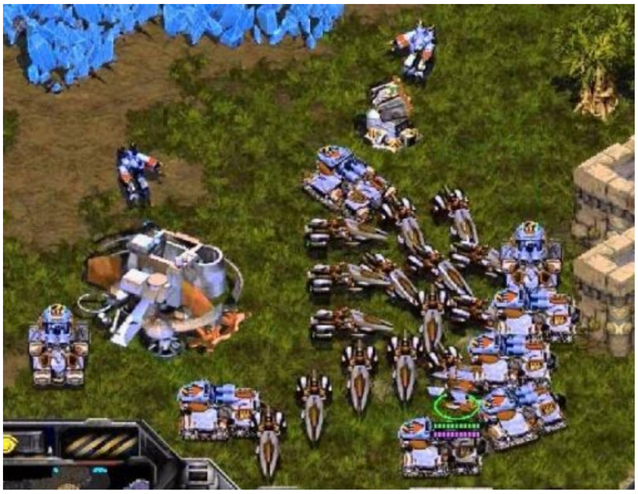

Fig. 2 Example of a main force (the strongest group of attack units).

main force. This is because each unit has a different attack power and range, and is subject to various factors that decrease power. Thus, it is necessary to integrate the contributions of all units in each location to understand the total attack power. We define a location's power as the sum of all units' contributions (total power). In some cases, the contribution of one strong unit is greater than the contributions of many weak units. A customized influence map is used to determine the positions of the main forces. Influence maps have been used widely in RTS research to calculate the attack force of an army [76] [77], where each unit's influence is calculated and integrated into a single value. The main force's position is simply determined as that with the highest influence. The unit's total influence considers the unit's influence power (IP), discount factor (DF), and influence range (IR); professional players determine these parameter values for each unit, considering each unit's characteristics (see Supplementary Materials). Each unit's influence is maximized in the area around it and decreases gradually with distance. The main force detection algorithm is described in Algorithm 1.

3.2 Categorization of Movement Directions for the Main Force

Due to the limited vision caused by the fog of war, a player can make no observations beyond their units and buildings. In the very early stages of the match, the players focus on their initial build orders (constructing buildings and producing units) and exploration of the game world using scouting units. After several minutes, the players start to produce attack units and develop new expansion bases to obtain more resources. It soon becomes important to manage many attack units as a single group to simplify the control tasks. Human players usually group attack units into a main force and several relatively small

Accepted manuscript

ACCEPTED MANUSCRIPT

Algorithm 1 Pseudo-code of the main force detection algorithm.

```
Require: array of units $P$
Ensure: position of main force
1:  Initialize influence map $M$, $n \times n$ matrix
2:  for unit in $P$ do
3:      for $i = 1, 2, ..., n$ do
4:          for $j = 1, 2, ..., n$ do
5:              $(x, y)$ = position of the unit
6:              set $IP$, $DF$ and $IR$ depend on type of unit
7:              $dist = distance((x, y), (i, j))$
8:              if $dist \leq IR$ then
9:                  $M(i, j) \mathrel{+}= IP - DF \times dist$
10:             end if
11:         end for
12:     end for
13: end for
14: position of main force = $\mathrm{argmax}_{k,l} M$
```

forces; it is not common to divide forces into equally sized armies. After the early build order stage, players start to control their main force as it moves into their opponent's area, returns to the home area, or stops at its current location. These strategic decisions are made based on all the different types of visual information available to the players. If it is beneficial, players can wait until an opponent is in close proximity to their base and simply accumulate attack units to build a large army. However, it is not always desirable to wait until opponents act because this allows them to perform many tasks without intervention. Thus, it is important to move the main force to the appropriate place at the right time. According to professional players, the movement direction of the main force can be categorized into six main types (Fig. 3), where the main force can stop at its current location, move to the opponent's area, or return to the home area. The opponent's area includes the main base, an expansion base, and their main force. The home area includes the main base and an expansion base.

The six movement actions for the main force are classified as:

- **Stop**: An absence of movement; the troops remain at their current position without receiving any commands.
- **Player's main base (PMB)**: Toward the allied main base; mainly applied when retreating from a battle or when the base is under attack.
- **Player's expanded bases (PEBs)**: To an area far away from the main base; applied when an expansion base is under enemy attack or needs more defensive forces.
- **Enemy's main force (EMF)**: Toward the location identified as the position of the EMF, for an attack; applied when the allied force is deemed superior and capable of beating the enemy's troops.

Accepted manuscript

ACCEPTED MANUSCRIPT

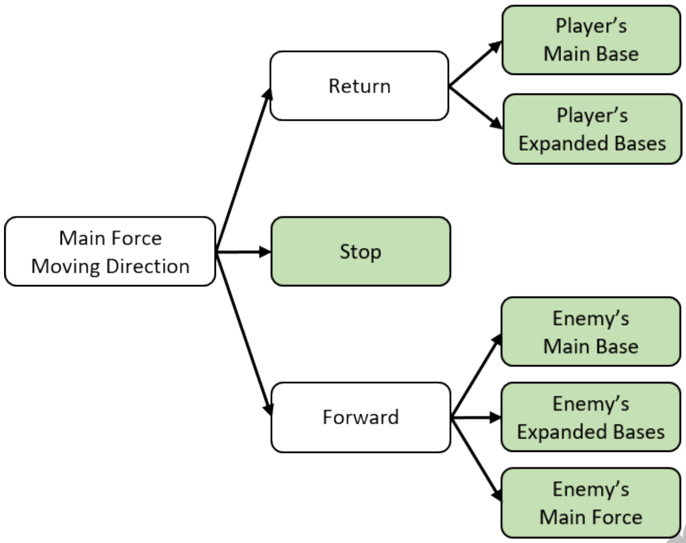

Fig. 3 Hierarchical categorization of the movement directions of the main force. Green nodes are labeled according to the task classification.

- **Enemy's main base (EMB)**: Taken when the EMF has either been forced to retreat, is positioned far away from their main base or, on rare occasions, for an assault on the base.
- **Enemy's expanded bases (EEBs)**: To attack an EEB that is far away from the EMB; typically applied when the EMF is positioned far from it.

3.3 Data Labeling for Human Replays

The replay files produced by human players contain all of the actions and game states that occur during gameplay. The duration of a match can range from several minutes to > 1 h. In each replay, it is possible to observe the decisions made by human players regarding the movements of their main force. We categorized these into the six groups delineated above. It is also possible to extract the locations of the main bases, expansion bases, and main forces for all of the players in the recorded match. Each player has only one main base and force, but the number of expansion bases can vary from zero to the maximum allowed by the map being played upon. Algorithm 2 describes the pseudo-code used to label each time step in the human replay data. The goal of the algorithm is to assign one of the six categories by analyzing the decisions made by human players regarding the target directions of the main force. In our experiment, MAX _TIME was set to 30 min and STEP was set to 5 s. According to the initial analysis, we found that clarifying the main force's target direction required ≥ 3 s. If the time was excessively short, the analysis could be hindered by noise due to physical movement delays or unexpected behavior by the units. If STEP was excessively long (e.g., 10 s), the analysis was

Accepted manuscript

ACCEPTED MANUSCRIPT

likely to miss changes in the decisions made by the human players regarding the main force's direction of movement.

Algorithm 2 Pseudo-code of the tactical intention labeling algorithm for human replays (the player's main force [PMF] corresponds to the Stop class).

```
Require: Replays $P$
Ensure: Labels
  for each replay do
      EMB = Find an enemy's main base
      PMB = Find a player's main base
      for $time = 1, 2, ..., MAX\_TIME$ do
          EEBs = Find all of an enemy's expanded bases
          PEBs = Find all of a player's expanded bases
          EMF = Find an enemy's main force
          PMF = Find a player's main force

          PMF_Units = Find all attack units around the PMF

          for each PMF_Unit do
              for each order assigned by Player from time to time+STEP do
                  Positions += unit → getOrderTargetPosition()[1]
              end for
          end for

          Target_Position = Average(Positions)
          Select Closest Location with the Target_Position
  from EMB, PMB, EEBs, PEBs, EMF, and PMF [2]
          Label the current time as the closest location
      end for
  end for
```

3.4 Deep learning for Decision Making

In StarCraft, there are three "races", and each player selects one for each match. The three races each have their own units and buildings. For experienced human players, it is natural to play better with, and thus favor, one of the three races. However, proficient players typically also play well with the other two races. Considering two-player matches, there are nine possible match combinations in the game because each player selects one of the three

[1] The getOrderTargetPosition() function retrieves the target position for the unit's order (from https://bwapi.github.io), which comprises the location to which the unit is trying to move or attack

[2] This function determines the Target Position based on all commands assigned to the units in the main force over a specific time (e.g., 5 s). It averages the locations where units are trying to move or attack

Accepted manuscript

ACCEPTED MANUSCRIPT

Table 1 Number of input channels in the CNNs and ViTs.

Player		Opponent		Terrain Channels	Total # of Channels
Race	# of channels	Race	# of channels		
Protos	15	Protos	15	3	33
Protos	15	Terran	16	3	34
Protos	15	Zerg	13	3	31
Terran	16	Protos	15	3	34
Terran	16	Terran	16	3	35
Terran	16	Zerg	13	3	32
Zerg	13	Protos	15	3	31
Zerg	13	Terran	16	3	32
Zerg	13	Zerg	13	3	29

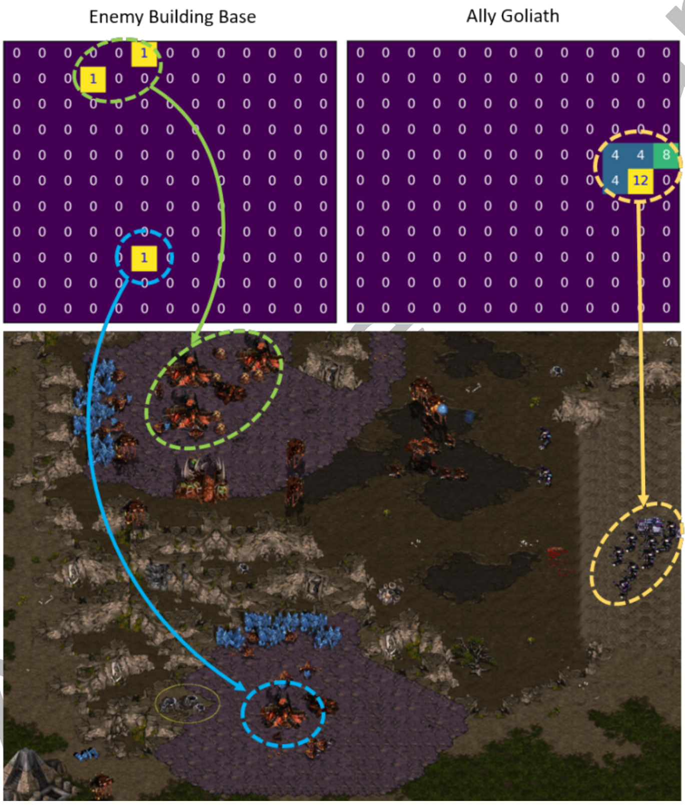

Fig. 4 Example input data for the CNNs and ViT: the "Enemy Building Base" channel, stores main buildings ("hatchery") for a Zerg player (upper left). The "Ally Goliath" channel, stores attack units ("goliaths") (upper right). The grid shows the relative locations of the units in the map. The numbers are the "influences" of units at each position.

Accepted Manuscript

ACCEPTED MANUSCRIPT

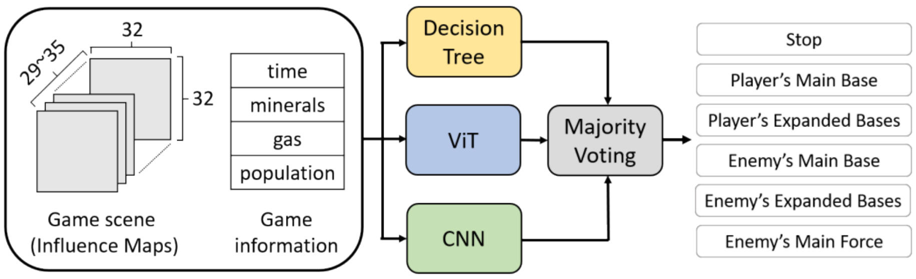

Fig. 5 Overall architecture of the proposed ensemble approach.

races; therefore, nine models can cover all match-ups (Table 1). The input data have 29-35 channels, with maps measuring 32 × 32 (the name of each channel is listed in the Supplementary Materials). For the unit channels, information about the position of a unit is received, and the influence of the unit is added to the value corresponding to its position on the map. In the case of enemy units, only units that have been seen at least once are added to the map. We used perfect information during the data preprocessing stage, the proposed ensemble model was trained and tested with imperfect information and did not require hidden information in either stage. The information about the enemy is imperfect because the fog of war restricts the visibility of enemy units and buildings, such that they are only visible when close to allied ones. This means that the information observed and fed into the model will be incorrect if the enemy moves their units or destroys buildings. The channels after unit one store information about the buildings built by both players. The first building channels record the positions of the main buildings used for storing resources for each race. And the second building channels record the buildings that the enemy's units can reach and attack. Finally, the third building channels store the details of all buildings other than the main and at-risk ones. The last three channels contain details of the terrain on the map; for each position, flat, hill, and blocked (except for airborne units) areas are represented. Fig. 4 shows examples of channels inputted to the model, where the "Enemy Building Base" channel (upper left) stores information concerning the presence of main bases (Hatchery), while the "Ally Goliath" channel (upper right) stores the influences of the attack unit (Goliath).

We built an ensemble model using three models: CNN, ViT, and decision tree. The majority voting rule is used to ensemble these models. The visualization of overall architecture is shown in Fig. 5.

For our CNN architecture, we used ResNet 18 [78] and fed the output to three fully connected layers, which consisted of 1,024, 256, and 6 neurons; the last layer used Softmax as its activation function. Moreover, the network used rectified linear units as the activation function for the convolution and fully connected layers. We used the cross-entropy loss function; the mini-batch size was 2,048. Dropout ($p = 0.5$) was applied to the fully connected layers. The CNN was optimized using the Adam optimizer. The learning rate depended on

Accepted manuscript

ACCEPTED MANUSCRIPT

the validation loss, and it was reduced when improvement ceased. If there was no improvement over 10 epochs, the learning rate was reduced by the decay rate; the initial learning rate was 0.0001, whereas the initial decay rate was 0.3.

Our ViT architecture contains three Transformer blocks followed by two fully connected layers with 1,024 and 6 neurons, respectively. The Transformer block contains an 8 × 8 patch, 1,024 hidden dimensions, and 16 attention heads. For the CNN, we used the cross-entropy loss function; the mini-batch size was 2,048. Dropout ($p = 0.5$) was applied to the fully connected layers. The Adam optimizer was used for optimization. However, the learning rate was not scheduled; it was fixed at 0.0001 for all training.

The CNN and ViT inputs are images. The size of the input channel is 32 × 32, and each set of data comprised 29–35 channels. The size of the StarCraft map is 128 × 128 but human players typically view this map by breaking it into smaller sections, so we used a map size of 32 × 32. This reduced size also had the advantage of decreasing the amount of input data. A vector of size 4 (game time, minerals, gas, and population) was directly fed to the fully connected layers of the CNN and ViT. These layers concatenated the image features and fed them to the fully connected layers.

The decision tree was the Classification and Regression Tree (CART) algorithm included in Scikit-learn. We used the Gini impurity to measure the quality of each split. Unlike a CNN and a ViT, a decision tree cannot accept image inputs. Thus, we flattened each image input to a vector and fed this to the decision tree. The data used to train the decision tree were therefore identical to the data used to train the other models, but their shapes differed.

3.5 Comparative evaluation

For comparative evaluation, we applied several baseline methods to our task. Since our method is based on the supervision of human demonstrations, it is proper to use previous methods supervised with human replay data as baselines. However, most previous tactical decision-making methods [5] [6] are based on RL with no utilization of human data, which is inappropriate for baselines. Therefore, we use two similar supervised learning approaches proposed for the macromanagement problem as baselines rather than those methods. The first method utilizes Multi Layer Perceptron (MLP) to decide the next build action [42]. The second method uses a transformer along with CNNs (Transformer with CNN) [27] to predict the winner and build order. Both methods use human replays as training data. On the other hand, our approach is based on the ensemble method [79]. Since our approach adopts the ensemble method, other ensemble methods should be used as baselines, such as meta-learning which is a model that makes predictions based on that of other models. Thus we trained two meta models, the decision tree and support vector machine (SVM). The meta model accepts y_c, y_t and y_d as inputs; it predicts the true label y, where y_c, y_t and y_d are the predictions of the CNN, ViT, and the decision tree, respectively. Finally, to evaluate the effect

Accepted manuscript

ACCEPTED MANUSCRIPT

of the ensemble, we also included component models of the proposed ensemble approach: CNN, ViT, and the decision tree.

4 Results and Discussion

For our experiment, we designed the proposed model using a well-known deep learning framework known as Torch, along with the well-known machine learning framework known as scikit learn.

4.1 Data Analysis

We downloaded StarCraft replays from two well-known game community portal sites, Ygosu and BWReplays, which include 1,408 replays of Korean professional and semi-professional players, and 10,649 replays where the player's actions per minute, (an important measure of a player's skill) were ≥ 250, respectively (Table 2). In total, the data comprised 12,057 replays of games played by experienced humans on various maps, with different match combinations of the Protoss, Terran, and Zerg races. While collecting replays, we made every effort to preserve the privacy of players to the highest possible extent. We only collected replay files and did not obtain any additional information that could identify the players.

Three million data samples were extracted at a sampling rate of 5 s and labeled as one of the six classes. Fig. 6 shows the main force detected in a scene from one match (resolution = 128 × 128); the supplementary video shows the main force detected throughout a full match. Our CNN model made predictions based on each sample and the accuracy was calculated as the ratio of correctly classified samples to all test samples.

To ensure that the replays were of matches played properly (i.e., without hacking or cheating), we used "minerals", one of the resources in StarCraft, as a criterion. For example, replays were not included if the mineral count was over 5,000 within 30 min. Typically, it is difficult to gather this many minerals while also constructing buildings and generating units. Data were excluded if the main force was not in the scene. At the beginning, only workers are available to collect minerals or gas, and some time is typically required to produce attacking units from buildings and resources. Thus, samples started from approximately 100 s and reached a maximum at 300 s (5 min), gradually

Table 2 Replay data collection (Sources: http://ygosu.com/replays and http://bwreplays.com). Both sources provide full match replays.

Site name	Number of	Proficiency	Note
Ygosu	1,408	(Semi-) Professional players	Full game replays
BWReplays	10,649	Amateur players over APM 250	Full game replays

Accepted manuscript

ACCEPTED MANUSCRIPT

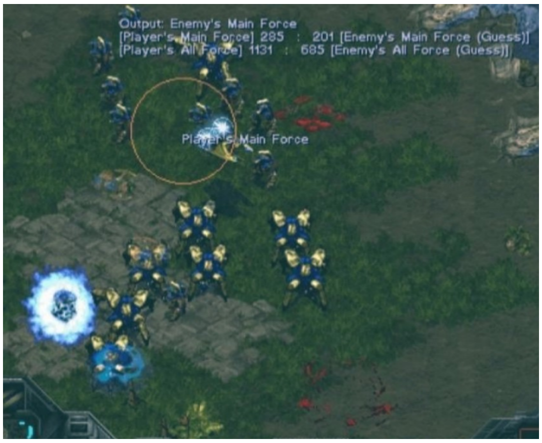
Output: Enemy's Main Force
[Player's Main Force] 285 : 201 [Enemy's Main Force (Guess)]
[Player's All Force] 1131 : 685 [Enemy's All Force (Guess)]

Fig. 6 Detection of a player's main force (circle) at a resolution of 128 × 128 (from supplementary video).

decreasing thereafter. Thus, most of the matches were played for at least 5 min and terminated at some point after this time.

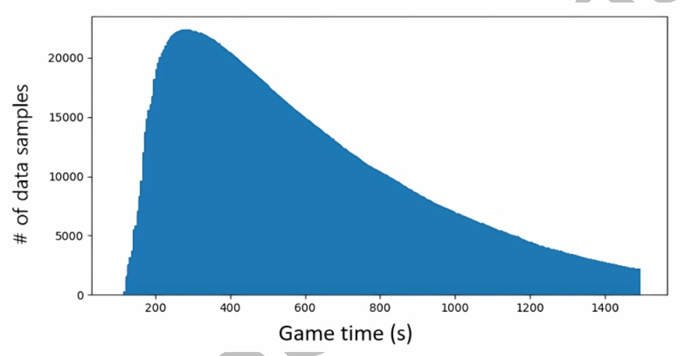

Fig. 7 Distribution of data samples according to match time

Figs. 7 - 9 show the basic statistics. The distribution of the six classes shows that the Stop class was the most common (approximately 25%). Fig. 10 shows the transition probabilities for the six movement decisions. For example, if the decision at time t was to Stop, the next decision was Stop 74% of the time, PMB 10% of the time, etc. Using the table, we constructed a simple predictor to estimate the next decision based only on the current one. For example, if the current decision was Stop, then the next decision was predicted as Stop. Using the statistics in the decision transition table, an accuracy of 49% was achieved.

If we simplify the decision-making problem, players may choose to continue or change the current movement direction for their main force. For the Stop

Accepted Manuscript

ACCEPTED MANUSCRIPT

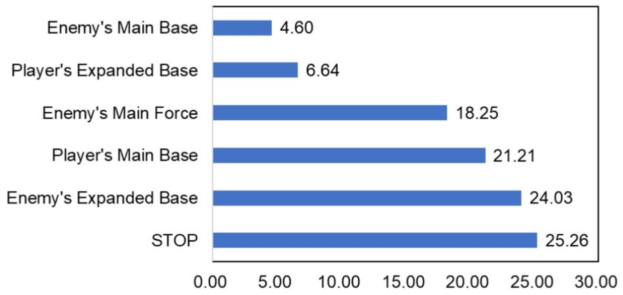

Fig. 8 Distribution of main force movement classes

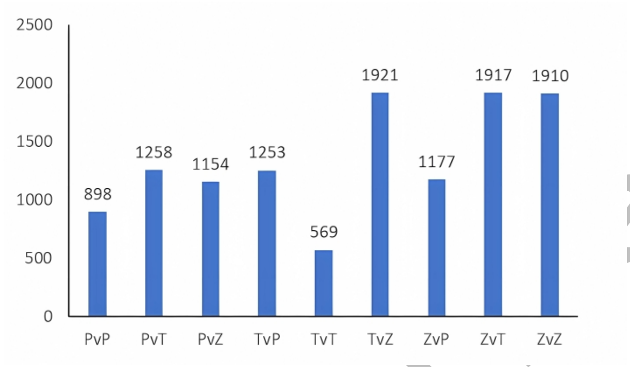

Fig. 9 Number of replays for each match type. P: Protoss, T: Terran, v: versus, Z: Zerg

decision, there was a high probability that the next decision was to continue with that action. Thus, the main force usually waited at a location for the player to make a decision, before moving to a target position with a probability of only $\frac{1}{4}$. When the main forces to the compared with all force was moving somewhere, stopping had a probability of approximately $\frac{1}{3}$-$\frac{1}{2}$, indicating that players frequently cancelled moves as the situation changed. The PMB → PMB sequence (40%) denotes a return to the main base, most often because of a need to defend it. The EMF → EMF sequence (32%) was likely to be the precursor to large-scale combat between the two main forces. Some sequences represent attacks, such as EMB → EEB (20%) and EMF → EEB (17%).

To explore how humans control the power of the main force, we calculated the ratios of the main forces to the compared with all units in terms of both the number of units and the sum of the influences. The average number ratio of the main force is $Ratio_t^N$ at t^{th} s, obtained by averaging the ratio of the main force at the t^{th} s of the i^{th} replay. Similarly, the average ratio of the main force in terms of influence is $Ratio_t^I$ at t^{th} s, obtained by averaging the ratio of the main force at t^{th} s of the i^{th} replay. The exact formula is as follows:

$$Ratio_t^N = \frac{\sum_i (N_{i,t}^{main}/N_{i,t}^{total})}{K}$$

$$Ratio_t^I = \frac{\sum_i (I_{i,t}^{main}/I_{i,t}^{total})}{K}$$

Accepted Manuscript

ACCEPTED MANUSCRIPT

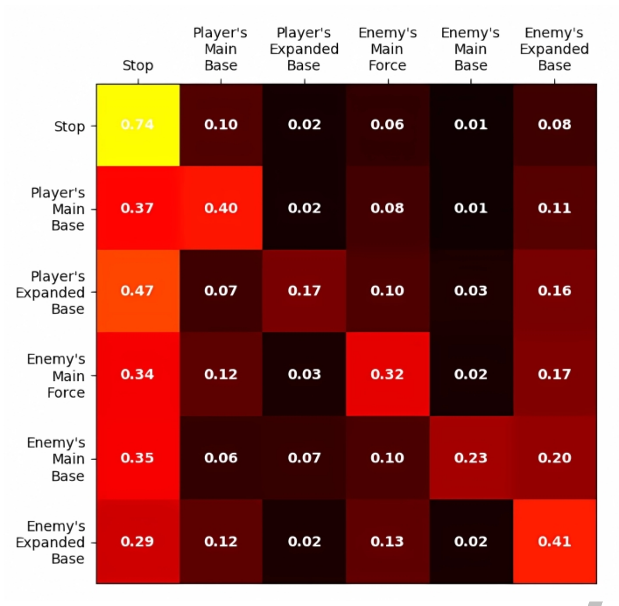

Fig. 10 Transition probability table (x-axis: time t+5 s, y-axis: time t) based on the ground truth data.

While $N_{i,t}^{main}$ and $N_{i,t}^{total}$ are the numbers of the main force units and all units respectively, at t^{th} s of the i^{th} replay. Where $I_{i,t}^{main}$ and $I_{i,t}^{total}$ are the summed influences of the main force units and all units respectively, at t^{th} s of the i^{th} replay. K is the total number of replays.

The $Ratio_t^I$ and $Ratio_t^N$ are shown over time in Figs. 11 and 12, respectively. In the early stage, both ratios are near 1, which means that only one force exists (most units belong to the main force). Both ratios decrease sharply until approximately 250 s, then decrease slowly thereafter. It is noteworthy that the ratio of the number of units (Fig. 12) is above the ratio for the influence (Fig. 11) after 250 s, but similar before then. This is because, in the early stage, most units are workers, and so they have similar influences. Subsequently, however, various battle units have been produced, so the unit influence becomes more diverse. Because strong units (which have high influence) typically belong to the main force, the $Ratio_t^I$ is higher than the $Ratio_t^N$. Generally, after the early stage of a match, approximately 40% of units belong to the main force and account for approximately 60% of the influence. This indicates that managing units using one strong main force is a common strategy for human players, and thus the main force plays an essential role in tactics.

4.2 Learning and Evaluation of the Deep learning model

Tables 3-5 list the test accuracies of the proposed model and the baseline models for the nine match types. We trained the nine ensemble models (CNNs,

Accepted manuscript

ACCEPTED MANUSCRIPT

Table 3 Top-1 test accuracy of the proposed model and baseline models. P: Protoss, T: Terran, Z: Zerg.

	Ensemble of CNN, DT, and ViT (proposed)	CNN	Decision tree (DT)	ViT (Vision Trans -former)	Meta decision tree	Meta SVM	MLP [42]	Trans -former with CNN [27]
P vs	**59.5**	53.3	52.9	55.1	52.9	52.9	48.5	47.8
P vs	**57.9**	53.4	51.2	52.6	51.2	51.2	44.3	39.3
P vs	**55.6**	49.8	50.9	50.8	50.9	50.9	41.8	36.4
T vs	**63.3**	56.4	57.4	58.8	57.4	57.4	46.7	37.9
T vs	**58.7**	52.3	55.5	51.5	55.5	55.5	45.5	33.2
T vs	**55.4**	48.6	49.8	50.3	49.8	49.8	41.6	35.5
Z vs	**54.1**	49.1	49.3	49.1	49.3	49.3	40.3	38.5
Z vs	**51.6**	44.3	46.5	46.1	46.5	46.5	40.8	35.5
Z vs	**55.7**	51.6	50.1	50.2	50.1	50.1	38.9	38.4
Avg.	**56.9**	51	51.5	51.6	51.5	51.5	43.2	38.1

Table 4 Top-2 test accuracy of the proposed model and baseline models. P: Protoss, T: Terran, Z: Zerg.

	Ensemble of CNN, DT, and ViT (proposed)	CNN	ViT (Vision Trans -former)	MLP [42]	Trans -former with CNN [27]
P vs	**79.9**	71.8	77.9	67.5	65.6
P vs	**76.8**	67.1	73.9	60.7	58.2
P vs	**74.9**	67.1	72.1	58.5	55.5
T vs	**81.4**	69	79.1	61.8	55.6
T vs	**77.3**	67.9	73.2	61.5	58.3
T vs	**76.5**	57.2	73.8	61.4	58.3
Z vs	**73.5**	67.1	70.3	57.7	59.1
Z vs	**72.2**	58.9	69.1	61.3	59.1
Z vs	**78.8**	73.4	75.9	61	64.5
Avg.	**76.8**	66.6	73.9	61.3	59.4

Table 5 Top-3 test accuracy of the proposed model and baseline models. P: Protoss, T: Terran, Z: Zerg.

	Ensemble of CNN, DT, and ViT (proposed)	CNN	ViT (Vision Trans -former)	MLP [42]	Trans -former with CNN [27]
P vs	**91.5**	85.6	91	79.	79.3
P vs	**87.9**	75.9	86.8	72.	72.5
P vs	**85.9**	78.8	84.6	71.	71.3
T vs	**90.6**	79.2	89.7	73.	74.5
T vs	**88.4**	77.1	86.8	75.	73.3
T vs	**88.4**	72.3	87.4	75.	73.8
Z vs	**85.4**	79.4	83.7	69.	74.5
Z vs	**85.7**	73.5	84.4	77.	76
Z vs	**92.9**	87.8	92.1	78.	85.1
Avg.	**88.5**	78.8	87.4	74.	75.6

Accepted manuscript

ACCEPTED MANUSCRIPT

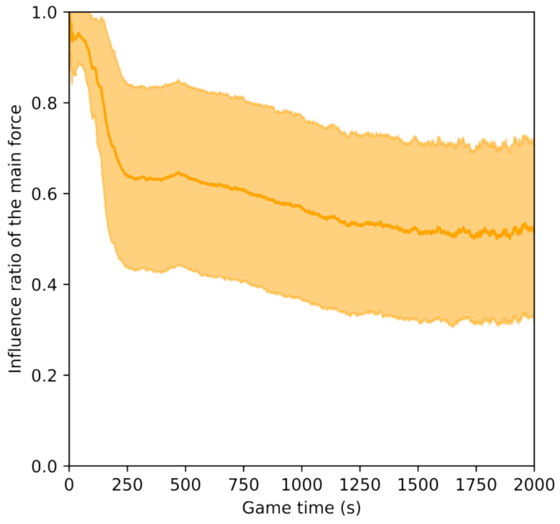

Fig. 11 Average ratios of the influences ($Ratio_t^I$) of units belonging to the main force compared with the influences of all units. The light yellow area is the standard deviation. The x-axis is game time (in s, *t*), and y-axis is the ratio.

a ViT, and a decision tree). Our model outperforms the baseline models in terms of all nine match types. The test top-1, -2, and -3 accuracies of the proposed model were 51.6–63.3%, 72.2–81.4%, and 85.4–92.9%, respectively. Among the nine match types, the proposed model for the Terran vs. Protoss match achieved the best top-1 accuracy for predicting the main force's direction 5 s later, at 63.3%. The performance of non-ensemble models (CNN, ViT, and decision tree) varied according to the match type. In the Protoss vs. Protoss match, the ViT outperformed the others; in the Terran vs. Terran match, the decision tree yielded the best result. The accuracies of the meta ensemble models (meta decision tree and meta SVM) are almost identical to the maximum accuracy of the three models (CNN, decision tree, and ViT) that form the ensemble. The two methods from previous works, MLP and the Transformer with CNN show relatively poor performance compared to other baselines. The average accuracies of MLP and the Transformer with CNN are 43.2 and 38.1, respectively, in contrast to the average accuracies of other baselines which are higher than 50. The average accuracies of the 7 baseline models over the nine match types were 51%, 51.5%, 51.6%, 51.5%, 51.5%, 43.2%, and 38.1% respectively. However, the proposed model, which is built through a simple ensemble technique (majority voting of three models), yields an average accuracy of 56.9%. In addition, in terms of the top-2 accuracy, the proposed model for the Terran vs. Protoss match performed best, with an accuracy of 81.4%. In terms of top-3 accuracy, the model achieved the highest accuracy for the Zerg vs. Zerg match, at 92.9%. Also throughout top-2

Accepted manuscript

ACCEPTED MANUSCRIPT

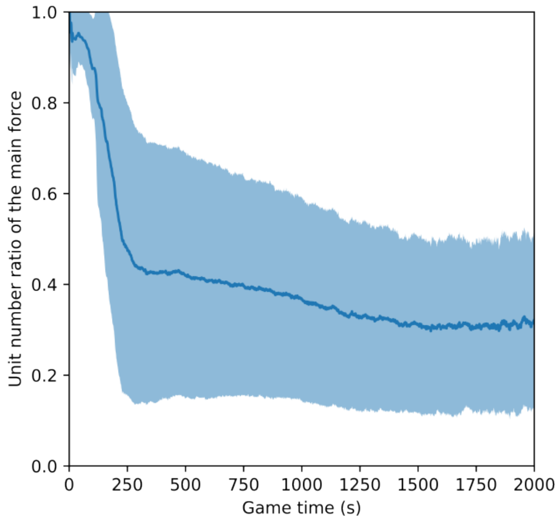

Fig. 12 Average ratio of the number ($Ratio_t^N$) of units belonging to the main force compared with the numbers of all units. The light blue area is the standard deviation. The x-axis is game time (in s, t), and the y-axis is the ratio.

and top-3 accuracies, the ViT shows high performance, close to the proposed method's accuracies. Because neither the decision tree nor the SVM delivered probabilities according to classes, we analyzed only the top-2 and -3 accuracies yielded by the proposed model, the CNN, and the ViT.

The phenomenon that the proposed ensemble model outperforms all baselines means that each module of the proposed ensemble model learned a different representation of the data. Each module has unique advantages and limitations. CNNs are specialized for image processing; they have strong inductive biases in terms of translation invariance and locality. CNNs thus perform well when image data are sparse. However, the inductive bias can disturb training if the data are large. The ViT exhibits less inductive bias and can thus learn unbiased representations if the data are adequate. Unlike the first two models, a decision tree performs poorly when fed high-dimensional data such as images. However, it is a simpler model. The combination processes data in a complex and diverse manner.

In a comparative view with similar existing methods (MLP[42] and Transformer with CNN [27]), our model shows the best performance. A further novel finding is that similar existing methods are worse than other baselines. The average top-1 accuracies of the other baselines are above 50%, but the MLP and the Transformer with CNN have top-1 accuracies approaching 40. This poor performance can be explained by their inappropriate network architecture. In the case of MLP, it shows poor performance since it is unsuitable for processing image data. However, although the Transformer with CNN is more

Accepted manuscript

ACCEPTED MANUSCRIPT

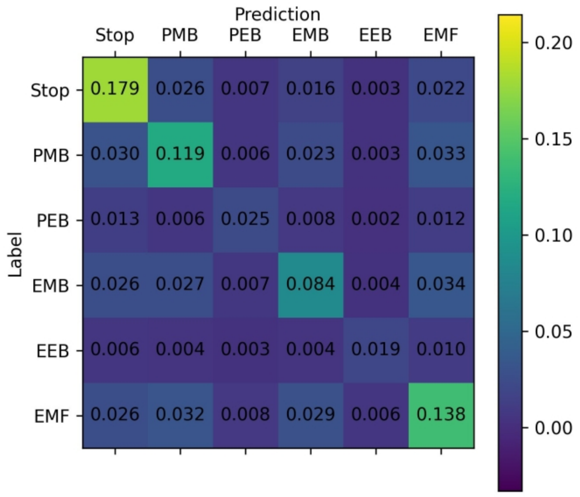

Fig. 13 Confusion matrix of the proposed model, for all test data samples of the nine match types. The number in each cell is the ratio of the sample number to the total number of samples. EEB: enemy's expansion base, EMB: enemy's main base, EMF: enemy's main force, PEB: player's expansion base, PMB: player's main base. The model often confuses non-Stop labels and Stop labels. Also, the model usually predicts Stop, PMB, EMB, and EMF correctly, but often confuses PEB and EEB.

Table 6 Inference time and number of parameters of proposed and similar existing models.

	Ensemble of CNN, DT, and ViT (proposed)	CNN	Decision tree (DT)	ViT (Vision Trans -former)	MLP [42]	Trans -former with CNN [27]
Inference time (ms)	24.91	16.9	0.17	7.84	1.63	35.42
# of parameters (million)	34	11	0	22	4	11

appropriate for processing image data, it shows a poorer performance than that of MLP. This may be explained by the excessively deep network of the Transformer with CNN. Since CNN and transformer sequentially process data, the network can be overfitted to the data.

Moreover, our approach focuses on predicting the future movements of the main force, while another approach aim to decide the destination of all armies [5]. Both approaches can have their advantages and disadvantages. For instance, in the early stages of a game, where the number of units is relatively

Accepted manuscript

ACCEPTED MANUSCRIPT

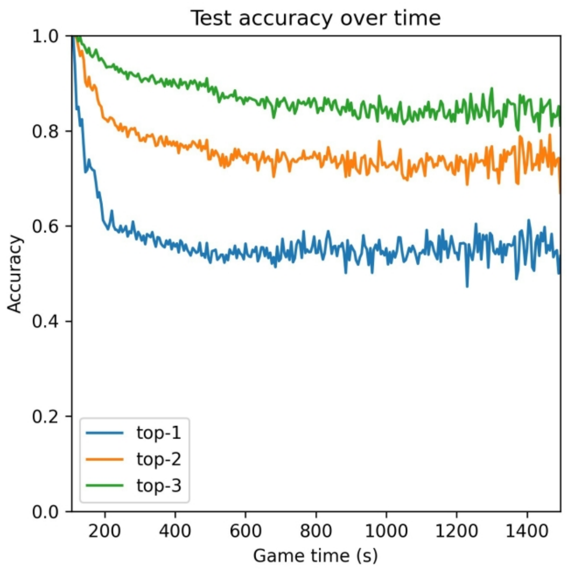

Fig. 14 Test accuracy of the proposed model for all test data samples over time.

small, predicting the movements of all units may be more feasible and useful. Nonetheless, as the game progresses and the number of units increases, it may become more practical and effective to focus on predicting the movements of the main force.

Although our method outperforms baselines, several limitations must be considered. First, our method is an ensemble approach that requires more computational resources during training and inferring. Table 6 shows inference times and the number of parameters of the proposed model and non-ensemble baselines. We produced this result using the NVIDIA V100 GPU. Except for the Transformer with CNN, the other four models were 8.01–24.74 ms faster than the proposed method. Also, the proposed model has 34 million parameters, which is larger than all baselines. However, this slow inference will not be a problem when applying it to real-time gameplay, because it can still infer more than 30 times in a second. In addition, unlike previous approaches for tactical decision making that have used RL [3] [4], our method is a supervised learning scheme. Thus, our approach requires a large amount of training data. Also, the performance of the trained model depends on the data quality. This is a double-edged sword because the performance can be easily improved using data of good quality but with poor quality data, the performance will deteriorate. Furthermore, since our approach simply imitates demonstrations from the data, the trained model is less likely to outperform the best human policy represented in the data. Future work can address this issue using the

Accepted manuscript

ACCEPTED MANUSCRIPT

self-supervised method [80] or offline RL [81] to make the model outperform human policies from the data.

In the case of meta-ensemble models (meta decision tree and meta SVM), they exhibited poorer performance, compared with the proposed majority voting approach. However, the SVM accuracy was almost identical to the maximum accuracy of the three models used in the ensemble (CNN, decision tree, and ViT). Because the meta models are trained using the same data, they appear to imitate the prediction of the model that was most accurate when using the training data.

The confusion matrix of the proposed model for all data samples is shown in Fig. 13. The proposed model typically predicted Stop, PMB, EMB, and EMF accurately, but it was correct less often for PEB and EEB. The model often confused labels of extended bases; the accuracies for both PEB and EEB are low. This result is explained by the uncertainties in base positions. Unlike the position of the main base, the positions of extended bases depend on the players, and there can be more than one extended base. Decisions related to such bases differ according to the players and their strategies. Therefore, from a modeling viewpoint, it is challenging to infer a human decision about an extended base. Also, the model often predicted non-Stop labels as Stop, reflecting label imbalance. As shown in Fig. 8, Stop is the most common label (25.26%).

Figure 14 shows the test accuracy for all data samples over time. In the early stage of a match, the accuracy is very high because most units are workers, and they typically stay near their base. It then decreases sharply due to the production of battle units, which are then organized into a new main force. After approximately 250 s, the accuracy gradually decreases. This can be interpreted as reflecting insufficient training data because the number of samples decreases over time, as shown in Fig. 7. This might also be caused by the simultaneous and gradual reduction in the ratio of the main force, as shown in Figs. 11 and 12.

4.3 Generalization over other RTS games

In this section, we discuss the potential challenges of applying the proposed approach to other RTS games. Although we implemented the proposed method on StarCraft, the proposed approach can be generalized onto other RTS games. To apply the proposed approach, three steps are required: collecting replays and extracting data from them, detecting the main force and labeling its future direction, and training the ensemble model using proper input encoding.

As most RTS games are repayable and use the available extraction software, collecting and processing the replays is typically straightforward. Considering the replays, the labels for the main force can be determined using the main force detection algorithm (Algorithm 1). However, this algorithm requires the parameters of each unit (IP, DF, IR), which the user must manually define based on the game. Finally, to train the model, the user must provide proper input encoding. This study encoded the game scene into a stack of each unit's

Accepted manuscript

ACCEPTED MANUSCRIPT

and building's influence maps, which were automatically generated based on their parameters. Therefore, the influence maps can be readily created after defining the parameters, and the user needs to select only the appropriate unit parameters to apply the proposed method.

Moreover, an essential consideration of applying this method to other RTS games is the significance of the main force in that game. As the Lanchester's laws are applicable to the RTS games, managing several units by clustering them into forces is a common strategy. Thus, the main force is likely to exist in the replays of the players. However, the effectiveness of the proposed approach may differ depending on the ratio of the main force in the game. In the analysis of the StarCraft replay data, approximately 40% of units pertained to the main force and account for approximately 60% of the influence. This suggests that the main force requires a substantial part of the military power, and its operation is a critical part of the game. However, the proposed method may be less effective in a game in which the ratio of the main force is less than that in the StarCraft. This is a potential limitation of the proposed approach, and a possible future direction is to address this issue by predicting the movements of the multiple forces. Another potential limitation is that the proposed method cannot handle tactical aspects that are not related to the main force. For instance, at the start of a game, a player might employ various tactics to disrupt their opponent's progress, such as harassing them with a scattered group of units as part of a "rush" strategy or deliberately obstructing the opponent's base development to force them to rethink their approach.

4.4 Applications

In this section, we discuss the mechanisms through which the proposed approach can improve the gameplay of RTS game agents. A potential application is to incorporate our approach as a module within AI bots, where the agent can determine the future direction of the main force based on the predictions of the ensemble model. Considering the fog of war, the model can be seamlessly integrated into the gameplay. The trained ensemble model can be directly used without additional training or be trained online while playing games to efficiently adapt to new games and strategies of the RTS agent. However, the online approach appears with the risk of overfitting. Applying our approach to the RTS game agents is an interesting avenue for future research.

Another possible application of the proposed approach is to predict the movements of the enemy main force. Although the proposed approach focuses on the allied main force, it can be applied to the main force of the enemy. To this end, the model needs to handle partial information, thereby creating challenges in training. The prediction techniques such as those presented in [44] reveal the fog of war can address this issue. Therefore, successful prediction of the movements of the main force of the enemy can enable the agent to infer the opponent's strategy and employ appropriate precautions.

Accepted manuscript

ACCEPTED MANUSCRIPT

5 Conclusion

Recently, StarCraft became an appealing environment for AI research because it provides varied and complex problems. Although tactics are an important part of RTS games, previous studies were based on heuristics and replays were not utilized for learning tactics.

In this work, we proposed a tactical decision-making method based on learning to control a main force using deep learning from replays of human games. We extracted approximately 3 million data samples from replays, then labeled them according to six categories defined by professional human players. Each data sample was converted into 29–35 channels as the input for our learning model. We used CNNs and ViTs—some of the most widely used techniques in deep learning research—to process this large among of data and its high-dimensional inputs. We built ensemble models using CNNs, ViTs and a decision tree to successfully guess the future movements of the main forces of players. We trained nine ensemble models to cover the different types of matches. The experimental results showed that the proposed method successfully estimated the six-class problem, with top-3, -2, and -1 accuracies of 92.9%, 81.4%, and 63.3%, respectively. The proposed model outperformed the baseline models for all nine match types in terms of the top-3, -2, and -1 accuracies. Also, we analyzed human decision-making processes related to the main force and showed that it accounts for approximately 60% of all units' influence and is thus essential for tactics. We formulated the main force detection problem according to the suggestions of professional human players, for whom handling the main force is critical throughout the match. Interpretation of the intentions of human players in the complex environments of RTS games can be very demanding. Traditional methods of learning encounter many difficulties when applied to these environments, due to the vast search space and endless number of possible actions. One advantage of our model is that replay data do not require complex pre-processing before the learning process.

We believe that our method could lead to new macro-level tactical decision-making problems in RTSs based on the perspective of human players. In addition, StarCraft AI research may benefit from the application of our model for building systems in terms of the build order, scouting, resource management, etc.

However, although our method successfully inferred human intentions when dealing with the main force, there is potential for improvement. Current image processing CNNs and a ViT do not perfectly learn human StarCraft tactics. Because neural networks are increasingly applied to various domains, they are being transformed, improved, and developed to suit such domains. In addition, although our analysis suggests that players have a single main force in most cases, however, there could be a case where the units are equally distributed among multiple forces rather than concentrated on a single main force. Considering such cases is one of the possible future improvements.

Future work should focus on the improvement of deep neural networks that learn human intentions in complex environments. One such example is

Accepted manuscript

ACCEPTED MANUSCRIPT

that of handling the relationships among forces. In this work, we only focused on the main force; however, strategy and tactics include setting the roles of forces and managing relations between them. In deep learning, graph neural networks (GNNs) are widely used for handling such relations through their strong inductive bias. GNNs have already been applied to micromanagement by focusing on the relations between units [82]. Thus, it may be useful for managing the relations among forces.

One direction for future work is the automation of the handcrafted parts of our approach. Currently, we determine each unit's parameters and categorize the main force movement based on heuristics, which may be suboptimal and may limit the performance of our approach. Automating these processes could lead to more efficient learning and better overall performance.

Another area for improvement is the consideration of the Rock Scissor Paper (RSP) relationships between units. Although our approach focuses on the main force, the composition of the enemy forces can greatly impact the effectiveness of the main force. For instance, a small cluster of certain units might be able to defeat a force of strong units, which could render the main force less impactful. By accounting for RSP relationships and other strategic considerations, our approach could be enhanced to better adapt to different game situations and improve its overall performance.

On the other hand, our method could be improved in a more practical way. In this work, we classified the destination of the main force into six categories. Instead of this, directly predicting the trajectory would allow for more continuous and flexible control for the main force. Furthermore, not only movement but also the composition of the forces is important in terms of strategy. Relations among forces may be affected by their unit composition. This point of view provides a path for future research. We used simple pre-processing methods based on timing and the amount of minerals to exclude non-informative data from the replays; however, this process must be modified to include a mechanism for identifying only the critical moments in matches (e.g., combat).

Supplementary information. Please see attached "Supplementary.pdf" and "video.wmv".

Acknowledgements. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government Ministry of Science and ICT (MSIT) (2021R1A4A1030075).

Data Availability. The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interests Authors have no conflicts of interest.

Accepted manuscript

ACCEPTED MANUSCRIPT

References

[1] Buro, M.: Real-time strategy games: A new ai research challenge. In: IJCAI, vol. 2003, pp. 1534–1535 (2003)

[2] Adil, K., Jiang, F., Liu, S., Jifara, W., Tian, Z., Fu, Y.: State-of-the-art and open challenges in rts game-ai and starcraft. Int. J. Adv. Comput. Sci. Appl **8**(12), 16–24 (2017)

[3] Synnaeve, G., Bessiere, P.: Special tactics: A bayesian approach to tactical decision-making. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 409–416 (2012). IEEE

[4] Soemers, D.: Tactical planning using mcts in the game of starcraft. Master's thesis, Maastricht University (2014)

[5] Lee, D., Tang, H., Zhang, J.O., Xu, H., Darrell, T., Abbeel, P.: Modular architecture for starcraft ii with deep reinforcement learning. In: Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference (2018)

[6] Xu, S., Kuang, H., Zhi, Z., Hu, R., Liu, Y., Sun, H.: Macro action selection with deep reinforcement learning in starcraft. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 15, pp. 94–99 (2019)

[7] Barriga, N.A., Stanescu, M., Besoain, F., Buro, M.: Improving rts game ai by supervised policy learning, tactical search, and deep reinforcement learning. IEEE Computational Intelligence Magazine **14**(3), 8–18 (2019). https://doi.org/10.1109/MCI.2019.2919363

[8] Farooq, S.S., Oh, I.-S., Kim, M.-J., Kim, K.J.: Starcraft ai competition report. AI Magazine **37**(2), 102–107 (2016)

[9] Čertický, M., Churchill, D., Kim, K.-J., Čertický, M., Kelly, R.: Starcraft ai competitions, bots, and tournament manager software. IEEE Transactions on Games **11**(3), 227–237 (2018)

[10] Ontanon, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game ai research and competition in starcraft. IEEE Transactions on Computational Intelligence and AI in games **5**(4), 293–311 (2013)

[11] Robertson, G., Watson, I.: A review of real-time strategy game ai. Ai Magazine **35**(4), 75–104 (2014)

[12] Olesen, J.K., Yannakakis, G.N., Hallam, J.: Real-time challenge balance in an rts game using rtneat. In: 2008 IEEE Symposium On Computational

Accepted manuscript

ACCEPTED MANUSCRIPT

Intelligence and Games, pp. 87–94 (2008). IEEE

[13] Andersen, P.-A., Goodwin, M., Granmo, O.-C.: Deep rts: a game environment for deep reinforcement learning in real-time strategy games. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–8 (2018). IEEE

[14] Buro, M.: Orts: A hack-free rts game environment. In: International Conference on Computers and Games, pp. 280–291 (2002). Springer

[15] Ontañon, S., Barriga, N.A., Silva, C.R., Moraes, R.O., Lelis, L.H.: The first microrts artificial intelligence competition. AI Magazine **39**(1), 75–83 (2018)

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

[17] Churchill, D., Buro, M.: Build order optimization in starcraft. In: Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (2011)

[18] Young, J., Hawes, N.: Evolutionary learning of goal priorities in a real-time strategy game. In: Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (2012)

[19] Garćıa-Sanchez, P., Tonda, A., Mora, A.M., Squillero, G., Merelo, J.: Towards automatic starcraft strategy generation using genetic programming. In: 2015 IEEE Conference on Computational Intelligence and Games (CIG), pp. 284–291 (2015). IEEE

[20] Kostler, H., Gmeiner, B.: A multi-objective genetic algorithm for build order optimization in starcraft ii. KI-Künstliche Intelligenz **27**(3), 221–233 (2013)

[21] Justesen, N., Risi, S.: Continual online evolutionary planning for in-game build order adaptation in starcraft. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 187–194 (2017)

[22] Huang, W., Yin, Q., Zhang, J., Huang, K.: Learning macromanagement in starcraft by deep reinforcement learning. Sensors **21**(10) (2021). https://doi.org/10.3390/s21103332

[23] Synnaeve, G., Bessiere, P.: A bayesian model for opening prediction in rts games with application to starcraft. In: 2011 IEEE Conference on Computational Intelligence and Games (CIG'11), pp. 281–288 (2011).

Accepted manuscript

ACCEPTED MANUSCRIPT

IEEE

[24] Cho, H., Park, H., Kim, C.-Y., Kim, K.-J.: Investigation of the effect of "fog of war" in the prediction of starcraft strategy using machine learning. Computers in Entertainment (CIE) **14**(1), 1–16 (2016)

[25] Cho, H.-C., Kim, K.-J., Cho, S.-B.: Replay-based strategy prediction and build order adaptation for starcraft ai bots. In: 2013 IEEE Conference on Computational Inteligence in Games (CIG), pp. 1–7 (2013). IEEE

[26] Dereszynski, E., Hostetler, J., Fern, A., Dietterich, T., Hoang, T.-T., Udarbe, M.: Learning probabilistic behavior models in real-time strategy games. In: Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (2011)

[27] Khan, M.J., Hassan, S., Sukthankar, G.: Leveraging transformers for starcraft macromanagement prediction. In: 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1229–1234 (2021). IEEE

[28] Churchill, D., Saffidine, A., Buro, M.: Fast heuristic search for rts game combat scenarios. In: Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (2012)

[29] Churchill, D., Buro, M.: Incorporating search algorithms into rts game agents. In: Eighth Artificial Intelligence and Interactive Digital Entertainment Conference (2012)

[30] Churchill, D., Buro, M.: Portfolio greedy search and simulation for large-scale combat in starcraft. In: 2013 IEEE Conference on Computational Inteligence in Games (CIG), pp. 1–8 (2013). IEEE

[31] Stanescu, M., Barriga, N.A., Buro, M.: Hierarchical adversarial search applied to real-time strategy games. In: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference (2014)

[32] Rathe, E.A., Svendsen, J.B.: Micromanagement in starcraft using potential fields tuned with a multi-objective genetic algorithm. Master's thesis, Institutt for datateknikk og informasjonsvitenskap (2012)

[33] Gabriel, I., Negru, V., Zaharie, D.: Neuroevolution based multi-agent system for micromanagement in real-time strategy games. In: Proceedings of the Fifth Balkan Conference in Informatics, pp. 32–39 (2012)

[34] Zhen, J.S., Watson, I.: Neuroevolution for micromanagement in the real-time strategy game starcraft: Brood war. In: Australasian Joint Conference on Artificial Intelligence, pp. 259–270 (2013). Springer

Accepted manuscript

ACCEPTED MANUSCRIPT

[35] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 4295–4304 (2018). PMLR

[36] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H., Kohli, P., Whiteson, S.: Stabilising experience replay for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 1146–1155 (2017). PMLR

[37] Young, J., Smith, F., Atkinson, C., Poyner, K., Chothia, T.: Scail: An integrated starcraft ai system. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 438–445 (2012). IEEE

[38] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., *et al.*: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

[39] Baek, I.-C., Kim, K.-J.: Web-based interface for data labeling in starcraft. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–2 (2018). IEEE

[40] Joo, H.-T., Lee, S.-H., Bae, C.-m., Kim, K.-J.: Learning to automatically spectate games for esports using object detection mechanism. Expert Systems with Applications **213**, 118979 (2023)

[41] Hu, Y., Li, J., Li, X., Pan, G., Xu, M.: Knowledge-guided agent-tactic-aware learning for starcraft micromanagement. In: IJCAI, pp. 1471–1477 (2018)

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

[43] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, / ., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[44] Jeong, Y., Choi, H., Kim, B., Gwon, Y.: Defoggan: Predicting hidden information in the starcraft fog of war with generative adversarial nets. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 4296–4303 (2020)

[45] Lin, Z., Gehring, J., Khalidov, V., Synnaeve, G.: Stardata: A starcraft ai research dataset. In: Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference (2017)

Accepted manusc

ACCEPTED MANUSCRIPT

[46] Lee, C.M., Ahn, C.W.: Feature extraction for starcraft ii league prediction. Electronics **10**(8), 909 (2021)

[47] Zhang, S., Wu, Y., Ogai, H., Inujima, H., Tateno, S.: Tactical decision-making for autonomous driving using dueling double deep q network with double attention. IEEE Access **9**, 151983–151992 (2021)

[48] Hoel, C.-J., Driggs-Campbell, K., Wolff, K., Laine, L., Kochenderfer, M.J.: Combining planning and deep reinforcement learning in tactical decision making for autonomous driving. IEEE transactions on intelligent vehicles **5**(2), 294–305 (2019)

[49] Watson, N., Hendricks, S., Stewart, T., Durbach, I.: Integrating machine learning and decision support in tactical decision-making in rugby union. Journal of the operational research society **72**(10), 2274–2285 (2021)

[50] Beal, R., Chalkiadakis, G., Norman, T.J., Ramchurn, S.D.: Optimising game tactics for football. arXiv preprint arXiv:2003.10294 (2020)

[51] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., *et al.*: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

[52] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.*: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)

[53] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937 (2016). PMLR

[54] Lample, G., Chaplot, D.S.: Playing fps games with deep reinforcement learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)

[55] Oh, J., Chockalingam, V., Lee, H., *et al.*: Control of memory, active perception, and action in minecraft. In: International Conference on Machine Learning, pp. 2790–2799 (2016). PMLR

[56] Ye, D., Chen, G., Zhang, W., Chen, S., Yuan, B., Liu, B., Chen, J., Liu, Z., Qiu, F., Yu, H., *et al.*: Towards playing full moba games with deep reinforcement learning. Advances in Neural Information Processing Systems **33**, 621–632 (2020)

Accepted manuscript

ACCEPTED MANUSCRIPT

[57] Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D.J., Mannor, S.: Learn what not to learn: Action elimination with deep reinforcement learning. Advances in Neural Information Processing Systems **31** (2018)

[58] Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. In: International Conference on Learning Representations (2018)

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

[60] Shin, Y., Kim, J., Jin, K., Kim, Y.B.: Playtesting in match 3 game using strategic plays via reinforcement learning. IEEE Access **8**, 51593–51600 (2020)

[61] Liskowski, P., Jaśkowski, W., Krawiec, K.: Learning to play othello with deep neural networks. IEEE Transactions on Games **10**(4), 354–364 (2018)

[62] Takada, K., Iizuka, H., Yamamoto, M.: Reinforcement learning for creating evaluation function using convolutional neural network in hex. In: 2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI), pp. 196–201 (2017). IEEE

[63] Stanescu, M., Barriga, N.A., Hess, A., Buro, M.: Evaluating real-time strategy game states using convolutional neural networks. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–7 (2016). IEEE

[64] Huang, J., Yang, W.: A multi-size convolution neural network for rts games winner prediction. In: MATEC Web of Conferences, vol. 232, p. 01054 (2018). EDP Sciences

[65] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)

[66] Irfan, A., Zafar, A., Hassan, S.: Evolving levels for general games using deep convolutional generative adversarial networks. In: 2019 11th Computer Science and Electronic Engineering (CEEC), pp. 96–101 (2019). IEEE

[67] Karavolos, D., Liapis, A., Yannakakis, G.N.: Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model. In: Proceedings of the 13th International Conference on the Foundations of

Accepted manusc

ACCEPTED MANUSCRIPT

Digital Games, pp. 1–10 (2018)

[68] Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: Pcgrl: Procedural content generation via reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 16, pp. 95–101 (2020)

[69] Vazifedan, A., Izadi, M.: Predicting human behavior in size-variant repeated games through deep convolutional neural networks. Progress in artificial intelligence **11**(1), 15–28 (2022)

[70] Ling, C., Tollmar, K., Gisslén, L.: Using deep convolutional neural networks to detect rendered glitches in video games. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 16, pp. 66–73 (2020)

[71] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=YicbFdNTTy

[72] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems **34**, 15084–15097 (2021)

[73] Xu, Y., Chen, L., Fang, M., Wang, Y., Zhang, C.: Deep reinforcement learning with transformers for text adventure games. In: 2020 IEEE Conference on Games (CoG), pp. 65–72 (2020). IEEE

[74] Ciolino, M., Kalin, J., Noever, D.: The go transformer: Natural language modeling for game play. In: 2020 Third International Conference on Artificial Intelligence for Industries (AI4I), pp. 23–26 (2020). IEEE

[75] Sopov, V., Makarov, I.: Transformer-based deep reinforcement learning in vizdoom. In: International Conference on Analysis of Images, Social Networks and Texts, pp. 96–110 (2022). Springer

[76] Sanchez-Ruiz, A.A., Miranda, M.: A machine learning approach to predict the winner in starcraft based on influence maps. Entertainment Computing **19**, 29–41 (2017)

[77] Uriarte, A., Ontanon, S.: Kiting in rts games using influence maps. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 8, pp. 31–36 (2012)

Accepted manuscript

ACCEPTED MANUSCRIPT

[78] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[79] Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1, pp. 1–15 (2000). Springer

[80] Ho, J., Ermon, S.: Generative adversarial imitation learning. Advances in neural information processing systems **29** (2016)

[81] Agarwal, R., Schuurmans, D., Norouzi, M.: An optimistic perspective on offline reinforcement learning. In: International Conference on Machine Learning, pp. 104–114 (2020). PMLR

[82] Shen, S., Fu, Y., Su, H., Pan, H., Qiao, P., Dou, Y., Wang, C.: Graphcomm: A graph neural network based method for multi-agent reinforcement learning. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3510–3514 (2021). IEEE

Accepted manuscript