IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 2, 2019, accepted July 13, 2019, date of publication July 25, 2019, date of current version August 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930581

An Uncertainty-Incorporated Approach to Predict the Winner in StarCraft II Using Neural Processes

MU LIN[1], TAO WANG[1], XIAOBO LI[1], JINJUN LIU[2], YANFENG WANG[1], YIFAN ZHU[1], AND WEIPING WANG[1]

[1]College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
[2]Academy of Military Science, Beijing 100091, China

Corresponding author: Tao Wang (rs023@163.com)

This work was supported in part by the National Social Science Foundation of China under Grant 14BTQ034.

ABSTRACT Real-time strategy (RTS) game is a kind of strategy game, in which the players compete for resources on 2D terrain by establishing the economy, training army, and guiding them into battle in real time. The winner prediction of the RTS games often involves studying a highly uncertain problem in an adversarial environment. In addition, the limit of the number of samples restricts on the application and performance of the prediction models. To obtain better winner prediction accuracy and maintain the prediction uncertainty under an adversarial environment, this paper proposes a neural network-based prediction method incorporated probability inference dealing with a small set of samples. This paper uses a dataset released based on SC2LE, a reinforcement learning environment released jointly by Blizzard Entertainment and DeepMind, and then employed the proposed neural processes model to build a winner prediction model. To verify, this paper implemented different features types' grouping and different game length grouping experiments for demonstrating better adaptability to such problems. Furthermore, this paper also implemented the SVM model experiments and compared the proposed method with the SVM model. Finally, when making predictions on a 1000 size testing data, the results show that the proposed prediction model achieves an accuracy of 0.811 at 200 and 0.821 at 1000 sizes of training sets, which is better than the SVM model with small training datasets.

INDEX TERMS Neural networks, prediction, uncertainty, machine learning, neural processes, real-time strategy game.

I. INTRODUCTION

Real-Time Strategy (*RTS*) is a sub-genre of strategy games, in which players compete on a 2-D terrain, by building a base, gathering resources, training units, and guiding them into battle in Real-Time [14]. The *RTS* game's prediction of outcomes (Win or Defeat) is an interesting area for Artificial Intelligence (AI) research. It is an effective environment, to conduct experiments for complex adversarial systems in *RTS* games [19], and the prediction of winners is done via depiction as a typical, large-dimensional, non-linear, probabilistic inference problem.

The main difficulty in predicting the winner arises from the requirement to research a large number of variables, in a wide, partially observable environment. *RTS* provides a game environment, with copious elements, for players to compete against each other. Players are expected to carefully adjust their strategies, based on a large number of observable dynamic variables, such as *resources*, *supplies*, *units*, and other factors, that will be referred to as features. When human players play the game, they would always have proficient knowledge on the contribution of each feature, which could increase their chances of winning the game, and would adjust their strategies accordingly [4]. However, for an AI, this would be an intensely challenging question. There are multiple complex interactions between features, that would lead to situations where changes in outcome cannot be expressed by linear functions. The unobservability of certain key features increases the complexity of the problem. A lot of features involved are completely random or unobservable, during the preceding game-play, which may have a major impact on a game's outcomes. Thus, the final result of the game is

The associate editor coordinating the review of this manuscript and approving it for publication was Sungroh Yoon.

often presented as a probability distribution, which depicts the Win or Defeat possibility of a game player. By observing the controllable variables, the game players can increase their chances in incurring one of the possible outcomes. However, there is still a probability that the exact opposite could happen.

The difficulty also arises from the fact that a player's strategy needs to be evaluated in an adversarial environment. Some units, in the game, have advantages against certain types of units, making certain strategies more effective against others. If the player happens to choose a strategy which restrains his opponent, he will have a higher probability of winning the game. Therefore, the observation of a single player's features is insufficient in making an accurate prediction for the game's outcome. Furthermore, in specific maps, certain strategies could be more effective than others, driving players to make different choices when facing different opponents and scenarios. All these components increase the difficulty of inference of game outcomes, and could cause the probability of the outcome of the game to become more uncontrollable.

According to the work carried out in this paper, we have considered that the key variables affecting the competition in the game are random and unpredictable. We have also made assumptions that there is a series of unobservable latent variables, which play a crucial role in the game's competition. We are unable to observe these latent variables directly, but can establish the relationship between observable variables and latent variables via a certain type of established model. Moreover, we have considered that the model needs to work in a Bayesian framework, to cope with the uncertainty of latent variables. Thus, the result of the evaluation needs to be a distribution function, rather than a specific value, for a situation where the given number of observations is limited. Many models can be used to establish the relationship between observable variables and implicit variables. For this paper, we chose a newly presented, neural network called Neural Processes (NPs) [6], [7], to cope with the limitations, and uncertainty issues. The Neural Processes, which combine the advantages of both the Neural Networks and the Gaussian process, can be used to analyze the uncertainty problem for smaller samples. As indicated in the following sections, this model can incorporate various variables of complex adversarial systems, and assist in providing a more convenient method for solving the problems of outcome predictions.

In order to achieve the above mentioned objective, the purpose of this paper is threefold: (i) to offer a novel approach for prediction in an *RTS* game, like *StarCraft* II, (ii) to apply the Neural Processes approach for analyzing a larger dimension of features in the game, from a publicly available dataset, and (iii) to discuss the potential benefits of Neural Processes approach towards the game's outcome prediction problems, by comparing several different models.

The structure of this paper can be divided as follows: Section 2 introduces the background needed for research, and presents certain models related to the subject's topics.

FIGURE 1. A screenshot of a *StarCraft* II match of DeepMind's AI, called AlphaStar, which use *Protoss* (one of the three optional races) to beat Team Liquid's pro player MaNa on Dec 19, 2018.

Section 3 improves the NPs model and training process. Section 4 addresses the case studies, the dataset, the network structure, and the experimental design. Section 5 captures the results of experimentation, and provides observations, based on the results. Section 6 provides the results, and discusses the performance of the proposed approach. Finally, Section 7 furnishes a conclusion for the entire study.

II. RELATED WORKS

A. STARCRAFT WITH LARGE-DIMENSIONAL FEATURES

After AlphaGo successfully beat the top human Go-player, in 2016, DeepMind announced that *StarCraft* would be its next challenge. *StarCraft* II has many characteristics, that are similar to complex adversarial systems, which includes partial observable information, adversarial uncertainty, decision-making in a dynamic environment, and a huge state-space – everything that makes it the perfect case study for an AI problem. Recently, DeepMind's program, AlphaStar, was able to defeat one of the professional *StarCraft* II players, with a score of 5-0, inspiring more research into AI for the *RTS* games.

A lot of case studies on AI, like the case of *StarCraft*, have been conducted during the past ten years. These works include the opening (first strategy) of opponents' prediction, automatic strategy generation, unit navigation, multiple units' cooperation, build order optimization, etc. The methods utilized include Bayesian model, Unsupervised Machine Learning, Behavioral Tree, Genetic-algorithm programming, and Case-based Reasoning (CBR) [2], [3], [5], [13], [15], [16], [21]–[23], [27].

The usage of large-dimensional features of *RTS* games, for prediction of the concerned issues in decision making, has become an active area of research. Usually, the dimension of features necessitates reduction, to make the predictions stable. The features can be separated into different levels for studies [14], such as the choice of opening strategy at a strategic level [20], or the timing involved in unlocking new technology, at the tactical level, or the information regarding the distance between units, in small-scale battles, at the reactive control level [17]. Weber and Mateas [26] focus on the timing aspects of a player's strategic decisions. They recognize the

opponent's strategy through feature-vectors, encoded at the time of the first production of units or buildings. Erickson and Buro [4] proposed a model using logistic regression, to evaluate the *RTS* game state, which could predict the winner for a given game's state, by reducing the estimation error of random variables, which in turn was done by establishing control over the variables. They divided the game's features into subsets of Economic, Military, Map Coverage, Micro Skill and Macro Skill, within different time intervals of the gameplay. Álvarez-Caballero [1] made use of 28 features to predict the winner, at a high accuracy level, without completion of the entire match.

B. DEALING WITH UNCERTAINTY USING A LATENT VARIABLE MODEL

Latent variable models [9]–[12], [29] offer a unified modeling approach for high-dimensional and uncertainty problems. It assumes that the randomness of the observed dataset D (sampled from a high-dimensional space $\mathbb{R}^d$) is determined by a latent variable z, in a low-dimensional space $\mathbb{R}^m$. Usually, the distribution $q(z)$ can be inferred by an observed dataset, through a statistical model, that connects the latent (unobserved) variables to observed variables [18]. Given the structure of the mapping model, the model parameters are calculated by maximizing the likelihood probability of observation data.

There are two ways to map the relationship between the observed data and latent variables, via the latent variable model – linear or nonlinear. A typical linear Latent variable model for Probabilistic Principal Component Analysis (PCA) was proposed by Tipping and Bishop [24], and its nonlinear form is the Gaussian Process Latent Variable Model (GPLVM), proposed by Lawrence [9]. In the field of machine learning, the variational autoencoder (VAE) [8] utilized latent variables for learning, and made remarkable progress in recent years. VAE can flexibly train nonlinear functions using neural networks, and is more proficient when compared to GPLVM, in its implementation efficiency.

C. PREDICTION USING NEURAL PROCESSES

By employing the observable features and unobservable implicit variables z, we can ascertain an approach to establish the relationship between the two. The Neural Network has a natural advantage in establishing a nonlinear relation mapping. It has higher computational efficiency, and a better learning effect, when compared with other nonlinear models. Since we require the evaluation of the game's competition in a Bayesian framework, we need a probabilistic approach to modeling the problem.

The Neural Processes model is a learning method, to represent distributions over functions, based on neural networks [6], [7], which can provide the nonlinear functional relationship between variables. NPs model captures the global uncertainty of a stochastic processes $F : \mathcal{X} \rightarrow \mathcal{Y}$ via a global latent variable z, which is represented by a parameterized probability distribution $p(z)$. For observed data pairs (x_i, y_i) with $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$, it learns the conditional distribution $p(z|x_i, y_i)$, and predicts y_i^* by querying the model with different x_i^* values and sampled z from $p(z|x_i, y_i)$. Since the latent variable z is a random variable, the predicted y_i^* is not a fixed value, but a sample from the distribution $p(y_i^*|x_i^*, z)$. NPs model can learn a distribution over a family of functions, which is very similar to the idea of the Gaussian process, and hence, is termed the Neural Process.

The Neural processes model is a black-box model, implying that it is suitable for building an effective prediction model, when the model structure between the observation variables and prediction results is unclear. When compared with the Bayesian Network model used to predict the outcomes of isolated battles [19], the Neural Process model is more direct and efficient in solving the analysis and prediction problems of complex adversarial systems, in a wide, partially observable environment.

III. FRAMEWORK: LATENT VARIABLE MODEL AND NEURAL PROCESSES MODEL

This section describes the model and analysis process used for the prediction of the outcome of the game. The *RTS* game outcomes prediction are required to solve the following problems: (i) how to determine the key variables that affect the winning or losing trend during the game, competing from a large dimension of observed variables, (ii) how to introduce the uncertainty of game competition into the model, including the scene complexity and the observable part of the opponent information, and (iii) how to solve the probabilistic inference problem of small samples.

A. A LATENT VARIABLES MODEL WITH UNCERTAINTY

Consider a set of match data, $M = \{m_1, ..., m_n\}$ extracted from a series of game replays. The match involves two players and has an end of two possible outcomes, i.e. one player or the other wins. Each m_i contains a group of explicit features x_i labeled with a game outcome y_i. Our goal is to build a model that can predict the outcome y_i with an input of x_i. We assume that there is a random latent variable z that can capture the uncertainty of the prediction. If the dataset M is obtained under a group of similar conditions (e.g., the same *match-up* [20], same *map* or other conditions), it could be sure that the entire dataset M will share a global distribution $P(z)$, representing the uncertainty of the whole dataset.

Since z is unobservable, the only thing we can do is to estimate the posterior distribution $P(z|x_i, y_i)$ of z through x_i and y_i. Sometimes, the explicit features observed may not be enough to dig out all the latent variables. In this case, we can make an attempt by introducing additional observable variables x_a in order to figure out the latent variables as much as possible, so that we can decrease the uncertainty to a considerably more acceptable range. **Fig.** 2 shows the schematic diagram of the proposed model. Since the distribution of z implies the random variables from the observed features x_i to the game outcomes y_i, the outcomes prediction y_i^* can

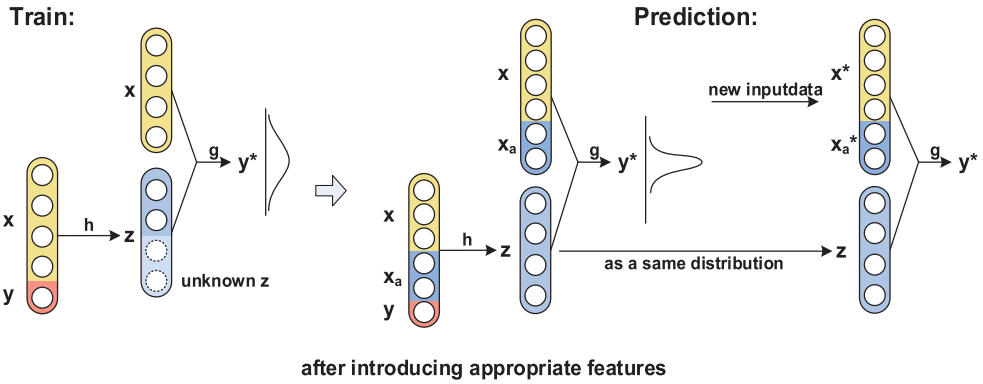


FIGURE 2. An overview of the proposed model. The training and prediction of the model use the same latent variable z to presents the uncertainty of game outcomes.

be modeled by $y_i^* = g(x_i, z)$ using some fixed, learnable functions [7].

Some latent variables of z represent the complex interaction between in-game resources. Every player should utilize the majority of the resources they possess during the game-play (economy, units, abilities, etc.) correctly in order to achieve the goal of winning the game. Even with the same resources, the scheduling of resource utilization by different players will lead to completely different outcomes. The resource utilization mode usually causes a non-linear impact on game outcomes, then it requires a nonlinear model to analyze the relationship between them. Other latent variables of z represent a large number of unknown variables that have yet to be observed but could directly be related to game competition, such as the opponent's unit combination and utilization mode. These variables cannot be fully observed but can be inferred by evaluating the representations in the game.

In the model above, the predicted output y^* is defined as a continuous value between [0, 1] instead of the discrete values 0 or 1. In this way, based on the theory of neural network, the deviation between the predicted game outcomes y^* and the actual outcomes y can be regarded as the optimization target required to train the network.

B. A VARIATIONAL INFERENCE METHOD

The vector z in **Fig.** 2 is sampled from a condition distribution of $P(z|x, y)$, which is difficult to calculate directly. Since the distribution is parameterized by a neural network, it could be solved by a variational inference method with the NPs model, just like VAE [8]. Specifically, it uses a simple distribution $Q(z|x, y)$ approximately approaching $P(z|x, y)$, and describes the deviation between them using KL-divergence:

$$KL\left[Q(z|x,y)||P(z|x,y)\right] = \mathbb{E}_{Q(z|x,y)}\left[\log Q(z|x,y)\right] - \mathbb{E}_{Q(z|x,y)}\left[\log P(z|x,y)\right]. \quad (1)$$

The variational lower-bound (ELBO) is given by following [7]:

$$\log P(y|x) \geq \mathbb{E}_{Q(z|x,y)}\left[\sum_{i=1}^{n} \log P(y_i|z,x_i) + \log \frac{P(z)}{Q(z|x,y)}\right]. \quad (2)$$

Here, a special training method is introduced to the NPs model. Given the observation set $M = (x_i, y_i)_{1:n}$, it randomly splits the observation set into a context set $(x_c, y_c)_{1:m}$ and a target set $(x_t, y_t)_{m+1:n}$ in multiple times, generating various data to train the neural network. The learning objective is using (x_c, y_c) and x_t to predicte y_t. By this training method, the NPs model uses $Q(z|x_c, y_c)$ to approximate the posterior distribution $P(z)$, so the ELBO is expressed as:

$$\log P(y_t|x_{c,t}, y_c) \geq \mathbb{E}_{Q(z|x_{c,t},y_{c,t})}\left[\sum_{t=m+1}^{n} \log P(y_t|z,x_t) + \log \frac{Q(z|x_c,y_c)}{Q(z|x_{c,t},y_{c,t})}\right]. \quad (3)$$

The approximated $Q(z|x_c, y_c)$ and $Q(z|x_{c,t}, y_{c,t})$ are assumed subjecting to Gaussian distribution $N(\mu_c, \sigma_c)$ and $N(\mu_{c,t}, \sigma_{c,t})$, making the loss function to be solved easily.

C. MODEL STRUCTURE AND PROCESS

We refer to the network architectures proposed by [7] in building our networks, which is structured in four components. (1) An **Encoder** h_r takes the input data and produces a representation vector $r_i = h_r(x_i, y_i)$ for each pair of match data (x_i, y_i). (2) An **Aggregator** a summarises the encoded inputs r_i to obtain a single global representation r_s, simply using the mean function $r_s = a(r_i) = \frac{1}{n}\sum_{i=1}^{n} r_i$. (3) An **Encoder** h_z produces a vector of $d_z \times 2$ dimension containing $\mu(z)$ and $\sigma(z)$. (4) A **Conditional decoder** g takes inputs of sampled global latent variable z as well as x_t and outputs the prediction value y_t.

To understand the process of our model, the procedure of the proposed prediction approach, which includes five steps, is shown in **Fig.** 3. The detailed steps are introduced step-by-step as follow:

1) **Step 1:** the entire data set is divided into a training set and a test set with sizes of N_{train} and N_{test}.
2) **Step 2:** before starting the training, a group of sub-sets S_i is set up as a data pool to receive data extracted from the training set. We firstly add a data sample of size N_S extracted randomly to the sub-sets S_i.

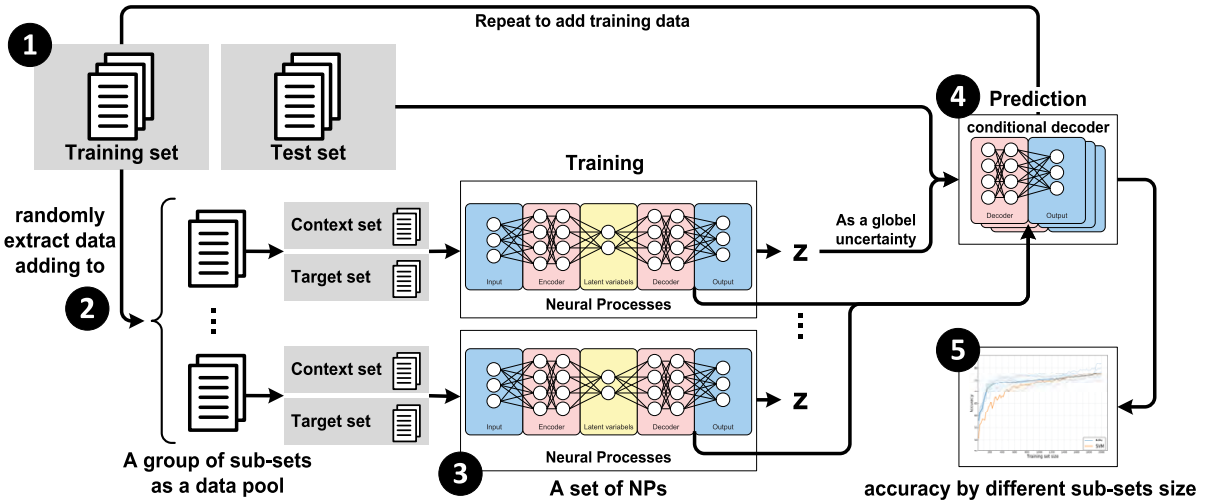


FIGURE 3. An overview of the training process. Since the split of the context set and target set is random, the parameters of the NPs network trained would be different even if the precisely same sub-set is given. Here we set up a set of neural processes to analyze the average accuracy of prediction results.

3) **Step 3:** a set of networks is trained with the input data in sub-sets S_i. The training is carried out with an epoch size of e_{train}. For each epoch, the S_i is randomly split into a context set and a target set with equal size, and is encoded by the encoder h_r and the aggregator a to obtain the parameters of a global latent distribution $Q(z|x_{c,t}, y_{c,t})$. To obtain a prediction at a target x_t, the model samples z from $Q(z|x_{c,t}, y_{c,t})$ and concatenate it with x_t, and map (z, x_t) through the conditional decoder g to obtain a sample from the predictive distribution of y_t. After training, record the parameters of $Q(z|x_{c,t}, y_{c,t})$ from each S_i, used for prediction later.
4) **Step 4:** the prediction process preserves the conditional decoder g. It samples z from $Q(z|x_{c,t}, y_{c,t})$, and then concatenates z with x_t^* in the test set and map (z, x_t^*) through g to a sample from the predictive distribution of y_t^*.
5) **Step 5:** here we would get a group of prediction samples of y_t^* for each S_i, and we summarizes the prediction accuracy results. Then, we add a new random extracted sample of size N_E to the subset S_i, and repeat the process starting from step 2.

IV. CASE STUDY

In this section, the case configuration used to verify the feasibility of the model described above is introduced. By using a public dataset, we want to determine whether the model above can establish a correlation between multi-dimensional features and the winning rate of the game in this public dataset, as well as how the latent variables work in predicting the winning rate. We configure the NPs network architectures according to the characteristics of the dataset and study the prediction effect of different feature sets through the comparison of the grouped experiments.

A. DATASET

There have been various datasets proposed by *StarCraft* over the last few years. SC2LE (*StarCraft* II Learning Environment [25] is a reinforcement learning environment based on the *StarCraft* II game released jointly by Blizzard Entertainment and DeepMind. Huikai Wu and Junge Zhang released a new dataset MSC based on SC2LE, which focuses on macro-management in *StarCraft* II [28]. The data set presented a series of playback eigenvector data through a standard process for SC2LE replays.

MSC provides two sub-datasets: a Global Feature Vector dataset and a Spatial Feature Tensor dataset, only the Global Feature Vector dataset is used in our model training. The dataset is formatted as a (T, M) matrix F, where F[t, :] is the feature vector for game frame t. Each row of F is an M-dimensional vector, with M varying as [RACE] v.s. [RACE]. Features which are used in our model are as follow:

1) **Reward**: i.e., the game outcomes. 0: *Defeat*, 1: *Win*.
2) **Cumulative Score**: Contains score, idle production time, idle worker time, total value units, total value structures, killed value units, and killed value structures, etc.
3) **Resources**: Contains various resources in game-playing, such as *minerals*, *vespene*, *supply cap*, *supply used*, idle worker count and army count, etc.
4) **Upgrades**: There are two types of upgrades that improve the functionality of player's units in *StarCraft* II.
   a. Numerical Upgrades refers to those upgrades which increase the attack and defense values of player's units, increasing their efficiency.
   b. Qualitative Upgrades affect how units function apart from their numerical combat effectiveness, which might for instance improvement of a unit. For example, *Personal Cloaking* enables *Ghosts* (a specialized infantry unit own by *Terran*) to use the Cloak ability, rendering a unit invisible to enemies unless it is revealed by detectors or effects.

MSC contains temporal information. In order to improve the network training speed, the original data is averaged according to the time dimension and the temporal information

is discarded. Once the mean is obtained, it is normalized. After normalizing the features, the data is represented in the form of a machine learning problem.

B. FEATURES

The features are grouped into two categories: Category A contains the Non-adversarial features, while category B contains the Adversarial ones shown in **Table** 1.

TABLE 1. Input features grouped by two categories.

Features	Remarks
Category A:	
idle production time	cumulative idle time of production
idle worker time	cumulative idle time of workers
total value units	value of units produced over entire game
collected resources	total *minerals* and *vespene* collected
spent resources	total *minerals* and *vespene* consumed
supply cap	a cap on how many units can be built
upgrades	percentage of time which an upgrade takes effect over the entire game
Category B:	
collection rate	collection rate of *minerals* and *vespene*
killed units value	units value player has killed
killed structures value	structures value player has killed
supply used	amount measured in supply how many units a player owns
army proportion	proportion measured in supply how many army units within player's total units

Category A's features provides an indication of the factors which are strongly associated with a player's own performance and weakly with the opponent. These features, including resources, upgrades and accumulated idle time represent the performance and strategy choices on the player's side. Category B's features indicate the competitive factors that exist between opponents and players. These types of features would change dramatically on the basis of engagement between the sides of the two players.

Some of the cumulative scores and resource features are included in category B because they are very adversarial; for example, the *killed units value* and the *killed structures value* are closely related to the combat situation of the two sides. However, the *resources collection rate* would frequently fluctuate as a result of the interference of the opposite side, making it more similar to the adversarial features. So, it is also included in category B.

C. NETWORK STRUCTURES

Fig. 4 shows the network structures with an input of both Non-adversarial and Adversarial features. The input to the neural network is an $N \times 751$ consisting of an N size data samples with 749 features. The data samples size N varies with the progress of training. The **Encoder** h_r is a 2-layer neural network followed by a ReLU function. The dimension d_r of r_i is predefined as same as d_z of z, less than the size of the feature set. The **Aggregator** a summarises the encoded inputs r_i using the mean function. The **Encoder** h_z is a 1-layer neural network, which outputs $\mu(z)$ by a aggregator and $\sigma(z)$ by a softplus function. The **Conditional decoder** g is a 2-layer neural network followed by a Sigmoid function. Here, we determine the dimension d_z according to the input features, as shown in **Table** 2.

TABLE 2. The dimension of input features and latent variable z.

Input features	Feature dimension	d_z dimension
only category A	405	2
only category B	344	2
both category A and B	749	4

V. THE PROBABILITY DISTRIBUTION OF PREDICTION

In this section, we present the results of our approaches for winner prediction by probabilistic inference.

As mentioned in section 3.1, the predicted output is defined as a continuous value between [0, 1] instead of discrete values. Thus, the Neural Processes model can give multiple predictions with a single inference by sampling the latent variables. In this way, by mapping the predicted value to a finite closed interval [0, 1], the probability distribution $p(y^*)$ of the outcomes can be estimated and the data can be analysed in a probabilistic way.

Take three samples as examples, as shown in **Fig.** 5. The predicted results are presented in a statistical bar chart. The figures (a) to (e) show the results of training for 100, 150, 200, 250, and 300 times. When the network training number is small, the predicted results are scattered between [0,1]. With an increase to the number of network trainings, the prediction results gradually become concentrated and gather near a certain value. We analyze the success rate of the prediction test set by taking the prediction result mean closer to 0 or 1 as the criterion of prediction result. Take the figure as an example, the red column represents the winning or losing situation of this sample, with 0 being negative and 1 being the winner. The bars of other colors represent the prediction results. Here, 100 samples of the hidden variable z are taken for each prediction, so each graph shows the statistics of 100 times of data.

The probability distribution $p(y^*)$ is a posterior distribution. When the Neural Processes make predictions, the distribution of the latent variable, z about the test set is unknown. Therefore, we cannot make predictions without prior knowledge of the latent variable z. Fortunately, we can use the network parameters obtained from the training set as the prior, assuming that the unknown uncertainties of the samples in the test set are similar to the training set. It works because the chosen datasets are all replays between *Terrans* (one of the three optional races in the game). By inputting z, sampled from the standard Gaussian distribution into the trained network, we can obtain the latent variables which conform to prior knowledge.

The distribution $p(z)$ represents the uncertain variables in the training set, but when the number of network training increases, this uncertainty may collapse. In order to

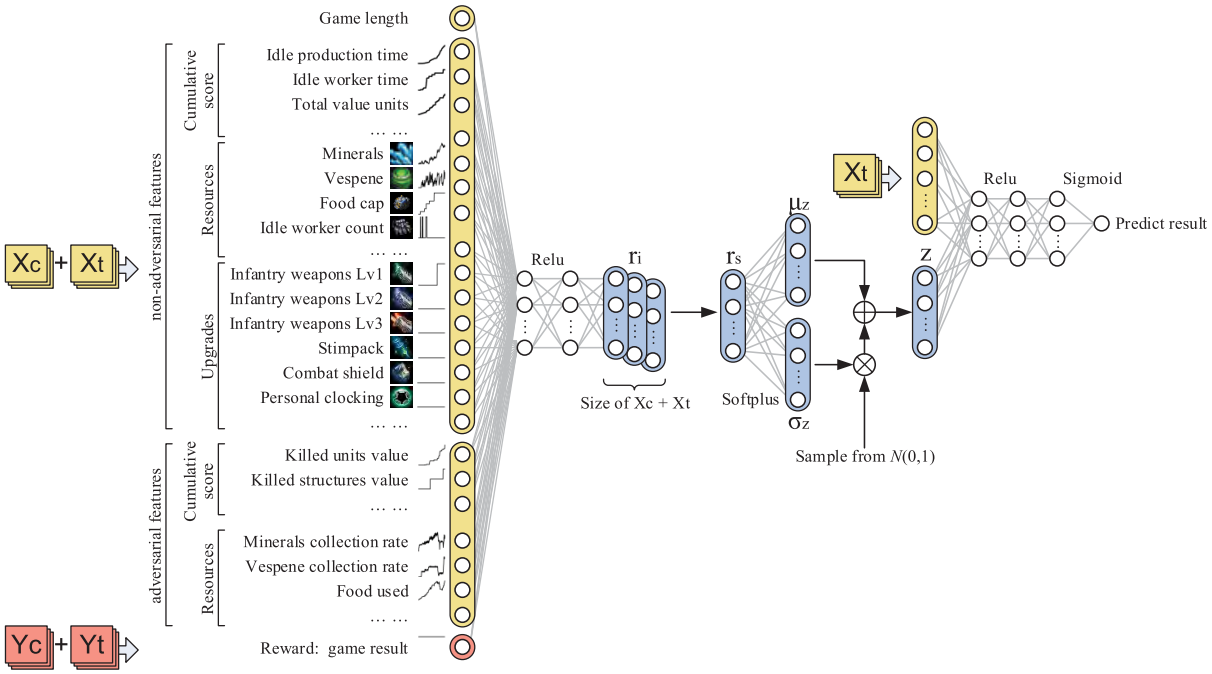


FIGURE 4. Neural Network Architecture with the input of *StarCraft* II features. The input layer consists of a vectorized state containing normalized mean values representing the features concatenated with game length and outcome. Only parts of features are shown on the diagram for clarity. The last layers are followed by an output layer using the sigmoid activation function, and the output of the network is the prediction of the game outcome.

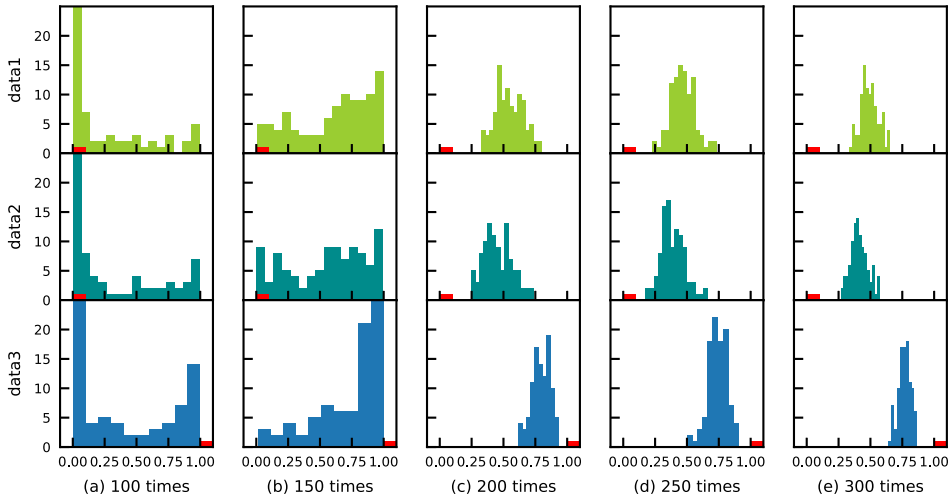


FIGURE 5. The prediction results in the form of histograms. There only gives three data samples to show the probability distribution of prediction.

keep the prediction uncertainty, the NPs trains network by randomly dividing the training set into ''context set'' and ''target set'' to generate new data. This approach has a potential benefit in that it is very effective when dealing with small datasets because the training set can be separated several times in order to generate enough samples.

Even so, with an increase to the number of trainings, the uncertainty will still a reduction. To maintain this uncertainty, we need to determine the appropriate number of training times. In the experiment, the posterior distribution of the predicted results is represented by a statistical histogram. By observing the dispersion degree of this posterior distribution, it can be decided when to stop the training. The Highest Posterior Density (HPD) interval is often used to describe the dispersion degree of the Posterior probability distribution. An HPD interval is the minimum interval containing a certain proportional probability density, such as 90%HPD or 95%HPD. Here, 90%HPD was selected as the index to evaluate the degree of dispersion of the posterior distribution.

TABLE 3. Prediction accuracy of the game outcomes compared NPs with SVM model.

Model	Features	Game length	Training set size									
			20	40	60	80	100	200	300	400	500	1000
NP	A	total	0.494	0.510	0.511	0.531	0.560	0.591	0.600	0.602	0.606	0.626
NP	B	total	0.562	0.646	0.686	0.739	0.764	0.811	0.818	0.815	0.822	0.821
NP	A+B	total	0.509	0.526	0.549	0.570	0.591	0.690	0.722	0.733	0.737	0.740
SVM	A	total	0.493	0.496	0.493	0.498	0.571	0.551	0.569	0.560	0.586	0.631
SVM	B	total	0.496	0.498	0.598	0.616	0.634	0.627	0.639	0.635	0.643	0.661
SVM	A+B	total	0.505	0.498	0.502	0.568	0.581	0.612	0.638	0.662	0.672	0.682
NP	A	short	0.542	0.563	0.594	0.616	0.612	0.640	0.646	0.654	0.658	0.668
NP	B	short	0.638	0.703	0.756	0.788	0.795	0.804	0.818	0.829	0.833	0.830
NP	A+B	short	0.521	0.533	0.537	0.558	0.583	0.688	0.762	0.768	0.767	0.769
SVM	A	short	0.501	0.594	0.524	0.527	0.530	0.567	0.644	0.687	0.686	0.705
SVM	B	short	0.501	0.501	0.501	0.615	0.737	0.768	0.807	0.780	0.812	0.834
SVM	A+B	short	0.501	0.527	0.598	0.648	0.682	0.758	0.779	0.797	0.800	0.840
NP	A	middle	0.542	0.530	0.562	0.587	0.612	0.621	0.628	0.628	0.633	0.635
NP	B	middle	0.643	0.732	0.763	0.781	0.797	0.805	0.809	0.812	0.809	0.815
NP	A+B	middle	0.563	0.660	0.677	0.678	0.683	0.734	0.765	0.767	0.769	0.769
SVM	A	middle	0.517	0.515	0.605	0.590	0.660	0.686	0.675	0.692	0.696	0.711
SVM	B	middle	0.485	0.717	0.766	0.752	0.753	0.803	0.816	0.817	0.816	0.833
SVM	A+B	middle	0.534	0.638	0.722	0.732	0.726	0.765	0.787	0.793	0.805	0.824
NP	A	long	0.495	0.511	0.516	0.517	0.520	0.538	0.539	0.547	0.555	0.568
NP	B	long	0.584	0.592	0.635	0.667	0.680	0.714	0.720	0.721	0.720	0.733
NP	A+B	long	0.527	0.536	0.550	0.572	0.583	0.625	0.638	0.645	0.654	0.675
SVM	A	long	0.499	0.533	0.507	0.496	0.520	0.567	0.563	0.570	0.584	0.614
SVM	B	long	0.490	0.490	0.498	0.626	0.624	0.655	0.659	0.661	0.664	0.675
SVM	A+B	long	0.547	0.578	0.575	0.624	0.653	0.660	0.659	0.671	0.683	0.710

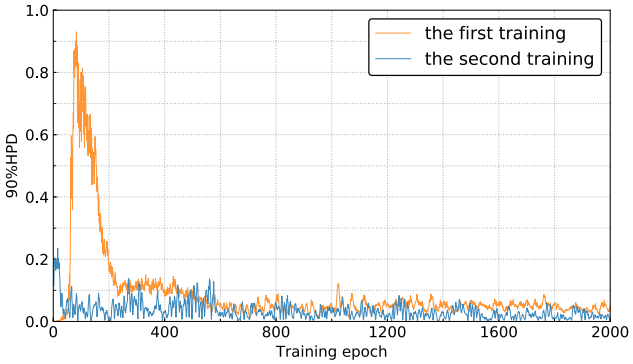


FIGURE 6. The 90%HPD curve of the first training on 20 training samples and the second training on 40 training samples.

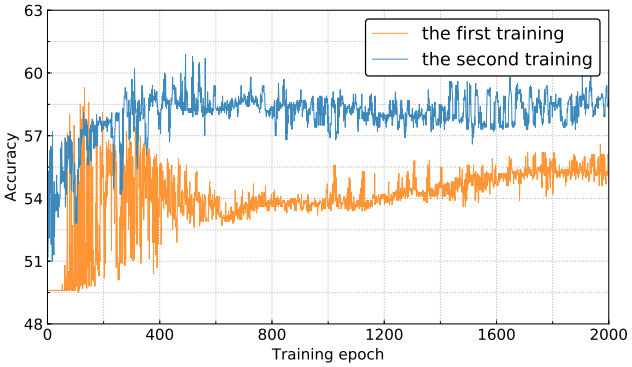


FIGURE 7. The accuracy curve of the first training on 20 training samples and the second training on 40 training samples.

Fig. 6 shows the 90%HPD of the prediction results with the increase of training epoch, and in **Fig.** 7 the accuracy with the increase of training epoch is shown.

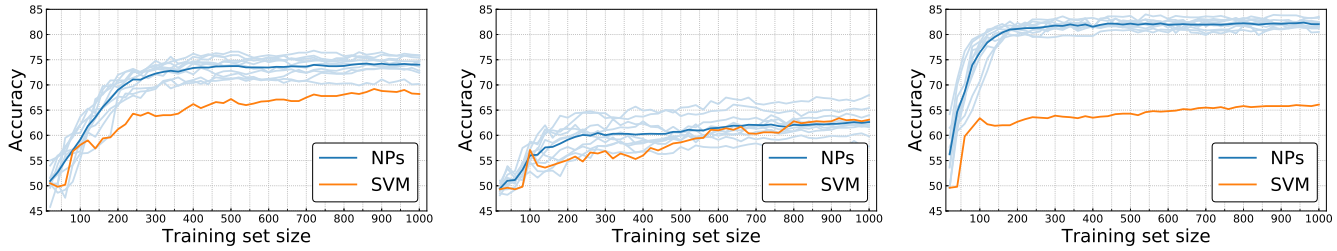
VI. EXPERIMENTAL RESULTS

In this section, we conducted two sets of experiments to study the impacts of features types and game length on prediction accuracy. The first set of experiments compares the accuracy of the predictions with different input features. The second set of experiments compares the effects of different game length on the prediction results. The winner prediction results are summarized in **table** 3.

A. ACCURACY ACROSS THE FEATURE TYPES

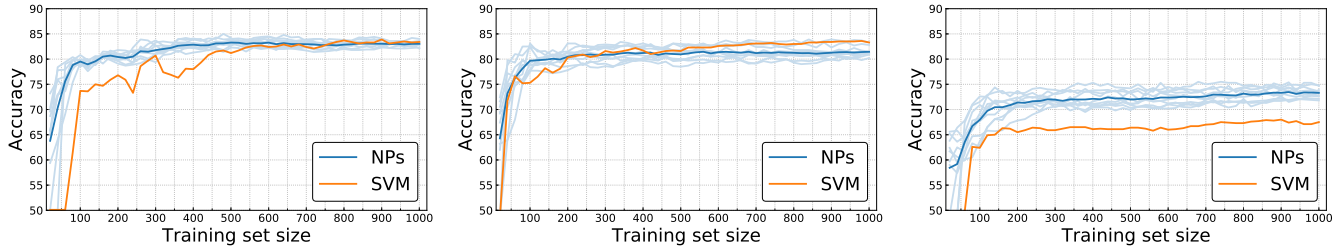
Experiments were performed in three groups. Group 1 takes only category A as inputs to train the network. Here, a total of $d_c = 405$ non-adversarial features is extracted from MSC dataset. Group 2 takes only category B as inputs, which has a total of $d_c = 344$ non-adversarial features. Group 3 takes both two categories features so that the input dimension of the network reached $d = d_c + d_e$. We are interested in how these two types of features help to capture latent variables and make more accurately predictions about the game outcome.

Table 3 also includes a simple SVM prediction models for comparison of prediction accuracy. In order to obtain stable



(a) features:A+B (b) features:A (c) features:B

FIGURE 8. The comparison results of prediction accuracy in the total data set by different features types of inputs. Due to the randomness of NPs network training, we give the prediction results by a set of NPs, and calculate their prediction accuracy mean value, which is compared with SVM.



(a) short matches (b) medium matches (c) long matches

FIGURE 9. The comparison results of prediction accuracy in different game length.

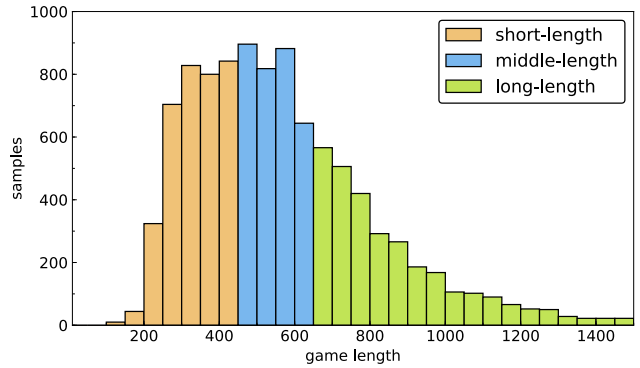


FIGURE 10. The distribution of game length of TvT replays in MSC dataset. The different length games would have a significant difference in the style of game-playing, so the parameters of the prediction model will also be different. We divided the replays into three groups according to the game length.

results, the mean accuracy of 10 predicted results was used as data for analysis, and the mean of the prediction accuracy on testing data is used for comparison.

Experiments were carried out for multiple times, and the size of training sub-sets increased by each time, from 20 to 1000. We are interested in how the training set size affected the accuracy of the prediction results of the test set. It is quite obvious that NPs model provides higher prediction accuracy at the training sets size are small.

Fig. 8 shows the change of the prediction accuracy with an increase of sub-sets size. As can be seen, the prediction accuracy with features A+B increased from 0.509 for 20 times to 0.740 for 1000 times. However, the accuracy rate was not significantly improved after the training set size reached about 300, which remained around 0.72. The accuracy of prediction results using only non-adversarial features or adversarial features as input is also similar.

B. ACCURACY ACROSS THE GAME LENGTH

Generally, players' strategies and tactics vary from the different phases in the *StarCraft* matches. Correspondingly, we should use different parameters of the network to predict game results. Through grouping the replays according to the length of replays, we can study the impact game length has on the prediction accuracy. We divided the data set according to the replay frame into three groups: a short-length, a middle-length, and a long-length group, as showen in **Fig.** 10. The short-length group contains the replays with the frames smaller than 450, the middle-length group with the frames of 450 to 650, and the long-length group the frames larger than 650.

The sample sizes of training set and test set for grouping are summarized in **table** 4. The prediction accuracy of the NPs and SVM models for the three grouped samples is given in **Fig.** 9. It can be seen that the prediction accuracy of the SVM model for short and middle length grouping is significantly improved compared with that of ungrouping, while the prediction accuracy of NPs model is not significantly improved after grouping. Considering that we have provided the game length as an input feature to the two models, such results show that the neural network structure adopted by NPs can better learn the impact of game length on the prediction results.

TABLE 4. The sample sizes of the training set and test set.

Game length	Training set size	Training sub-sets size	Test set size
total	8794	20 to 1000	1000
short	3534	20 to 1000	1000
medium	3258	20 to 1000	1000
long	3002	20 to 1000	1000

VII. CONCLUSION AND FUTURE WORK

In this work, a probability inference method for the prediction of game outcomes by *StarCraft* II replays was described. The NPs neural networks model was developed to estimate the winner of the game through adversarial features and non-adversarial features. The model is able to give a good prediction of game outcomes in the case of processing large-dimensional features data with uncertainty under an adversarial environment.

We compared the training dataset size requirements of the NPs model with the SVM model. The experimental results demonstrate that the NPs prediction accuracy is higher when training with small datasets. Moreover, the NPs can give a prediction accuracy rate of over 80% for data sets mixed with different lengths of game time, which is much better than the SVM model.

In future work, many extended researches could be developed as follows:

1) Based on feature selection, the proposed prediction method will get better results.
2) By replacing the mean value of features with dynamic data containing time dimension information, we can apply the model to predict the winner of game-playing in real-time.
3) With an improved classifier, a useful model could be developed for evaluating the strategies and situations of players. It can be used to improve the adaptability of game AI, making it more likely to choose the right strategy.

REFERENCES

[1] A. A. Caballero, J. J. M. Guervós, P. García-Sánchez, and A. F. Ares, "Early prediction of the winner in StarCraft matches," in *Proc. 9th Int. Joint Conf. Comput. Intell.*, Jan. 2017, pp. 401–406. doi: 10.5220/0006587304010406.

[2] N. A. Barriga, M. Stanescu, and M. Buro, "Combining strategic learning with tactical search in real-time strategy games," in *Proc. 13th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, Sep. 2017, pp. 9–15. [Online]. Available: https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15814

[3] G. Bosc, P. Tan, J.-F. Boulicaut, C. Raïssi, and M. Kaytoue, "A pattern mining approach to study strategy balance in RTS games," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 2, pp. 123–132, Jun. 2017. doi: 10.1109/TCIAIG.2015.2511819.

[4] G. K. S. Erickson and M. Buro, "Global state evaluation in StarCraft," in *Proc. 10th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)* Sep. 2014, pp. 1–7. [Online]. Available: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/view/8996

[5] P. Garćia-Sánchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo, "Towards automatic StarCraft strategy generation using genetic programming," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug./Sep. 2015, pp. 284–291. doi: 10.1109/CIG.2015.7317940.

[6] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami, "Conditional neural processes," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 1690–1699. [Online]. Available: http://arxiv.org/abs/1807.01613

[7] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," 2018, *arXiv:1807.01622*. [Online]. Available: https://arxiv.org/abs/1807.01622

[8] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Representations (ICLR)*, Dec. 2014, pp. 1–14. [Online]. Available: http://arxiv.org/abs/1312.6114

[9] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models," *J. Mach. Learn. Res.*, vol. 6, pp. 1783–1816, Jan. 2015. [Online]. Available: http://jmlr.org/papers/v6/lawrence05a.html

[10] N. D. Lawrence, "Learning for larger datasets with the Gaussian process latent variable model," in *Proc. 11th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Mar. 2007, pp. 243–250. [Online]. Available: http://jmlr.org/proceedings/papers/v2/lawrence07a.html

[11] S. Lipovetsky, "Latent variable models and factor analysis," *Technometrics*, vol. 43, no. 2, p. 111, 2001. doi: 10.1198/tech.2001.s568.

[12] S. Mohamed and B. Lakshminarayanan, "Learning in implicit generative models," 2016, *arXiv:1610.03483*. [Online]. Available: https://arxiv.org/abs/1610.03483

[13] I.-S. Oh and K.-J. Kim, "Testing reliability of replay-based imitation for StarCraft," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug./Sep. 2015, pp. 536–537. doi: 10.1109/CIG.2015.7317899.

[14] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game AI research and competition in StarCraft," *IEEE Trans. Comput. Intell. AI Games*. vol. 5, no. 4, pp. 293–311, Dec. 2013. doi: 10.1109/TCIAIG.2013.2286295.

[15] G. Robertson and I. Watson, "Building behavior trees from observations in real-time strategy games," in *Proc. Int. Symp. Innov. Intell. Syst. Appl. (INISTA)*, Sep. 2015, pp. 1–7. doi: 10.1109/INISTA.2015.7276774.

[16] K. Shao, Y. Zhu, and D. Zhao, "Cooperative reinforcement learning for multiple units combat in starCraft," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2017, pp. 1–6. doi: 10.1109/SSCI.2017.8280949.

[17] K. Shao, Y. Zhu, and D. Zhao, "StarCraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Trans. Emerg. Topics Comput. Intell.* vol. 3, no. 1, pp. 73–84, Feb. 2019. doi: 10.1109/TETCI.2018.2823329.

[18] A. Skrondal and S. Rabe-Hesketh, "Latent variable modelling: A survey," *Scand. J. Statist.*. vol. 34, no. 4, pp. 712–745, Dec. 2010. doi: 10.1111/j.1467-9469.2007.00573.x.

[19] M. Stanescu, S. P. Hernandez, G. Erickson, R. Greiner, and M. Buro, "Predicting army combat outcomes in StarCraft," in *Proc. 9th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)*, Oct. 2013, pp. 86–92. [Online]. Available: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/view/7381

[20] G. Synnaeve and P. Bessière, "A Bayesian model for opening prediction in RTS games with application to StarCraft," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug./Sep. 2011, pp. 281–288. doi: 10.1109/CIG.2011.6032018.

[21] G. Synnaeve and P. Bessière, "A Bayesian model for plan recognition in RTS games applied to StarCraft," in *Proc. 7th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)*, Oct. 2011, pp. 79–84. [Online]. Available: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE11/paper/view/4062

[22] G. Synnaeve and P. Bessière, "Multiscale Bayesian modeling for RTS games: An application to StarCraft AI," *IEEE Trans. Comput. Intell. AI Games*. vol. 8, no. 4, pp. 338–350, Dec. 2016. doi: 10.1109/TCIAIG.2015.2487743.

[23] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. A. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connect. Sci.*, vol. 26, no. 1, pp. 45–63, Mar. 2014. doi: 10.1080/09540091.2014.885279.

[24] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *J. Roy. Stat. Soc.*. vol. 61, no. 3, pp. 611–622, 1999. doi: 10.1111/1467-9868.00196.

[25] O. Vinyals *et al.*, "StarCraft II: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*. [Online]. Available: https://arxiv.org/abs/1708.04782

[26] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *Proc. IEEE Symp. Comput. Intell. Games (CIG)*, Sep. 2009, pp. 140–147. doi: 10.1109/CIG.2009.5286483.

[27] S. Wender and I. Watson, "Combining case-based reasoning and reinforcement learning for unit navigation in real-time strategy game AI," in *Proc. 22nd Int. Conf. (ICCBR)*, Sep. 2014, pp. 511–525. doi: 10.1007/978-3-319-11209-1_36.
[28] H. Wu, J. Zhang, and K. Huang, "MSC: A dataset for macro-management in StarCraft II," 2017, *arXiv:1710.03131*. [Online]. Available: https://arxiv.org/abs/1710.03131
[29] C. Yoo, "The Bayesian method for causal discovery of latent-variable models from a mixture of experimental and observational data," *Comput. Statist. Data Anal.*. vol. 56, no. 7, pp. 2183–2205, Jul. 2012. doi: 10.1016/j.csda.2012.01.010.

MU LIN received the B.S. degree from the College of Mechatronics Engineering and Automation, National University of Defense Technology (NUDT), Changsha, China, in 2006, where he is currently pursuing the M.S. degree with the College of Systems Engineering. His research interests include systems engineering and simulation, machine learning, and data mining.

TAO WANG received the Ph.D. degree in software engineering from the National University of Defense Technology (NUDT), Changsha, China, where he is currently an Associate Professor. His research interests include systems engineering and simulation, multi-agent decision making under uncertainty, and data mining.

XIAOBO LI received the Ph.D. degree in control science and engineering from the National University of Defense Technology (NUDT), Changsha, China, where he is currently an Associate Professor. His research interests include systems engineering and simulation, multi-agent decision making under uncertainty, and SoS engineering.

JINJUN LIU received the Ph.D. degree in military science from the Air Force Command College, Beijing, China. He is currently an Assistant Researcher with the Academy of Military Sciences. His main research interests include system engineering and system evaluation.

YANFENG WANG received the Ph.D. degree in military science from the Rocket Force Command College, Wuhan, China. He holds a Postdoctoral position with the National University of Defense Technology (NUDT). His research interests include systems engineering and simulation.

YIFAN ZHU received the Ph.D. degree in systems engineering from the National University of Defense Technology (NUDT), Changsha, China, where he is currently a Professor. His research interests include systems engineering and simulation.

WEIPING WANG received the Ph.D. degree in systems engineering from the National University of Defense Technology (NUDT), Changsha, China, where he is currently a Professor. His research interests include systems engineering and simulation.