DefogGAN: Predicting Hidden Information in the StarCraft Fog of War with Generative Adversarial Nets

Yonghyun Jeong, Hyunjin Choi, Byoungjip Kim, Youngjune Gwon
Samsung SDS

Abstract

We propose DefogGAN, a generative approach to the problem of inferring state information hidden in the fog of war for real-time strategy (RTS) games. Given a partially observed state, DefogGAN generates defogged images of a game as predictive information. Such information can lead to create a strategic agent for the game. DefogGAN is a conditional GAN variant featuring pyramidal reconstruction loss to optimize on multiple feature resolution scales. We have validated DefogGAN empirically using a large dataset of professional StarCraft replays. Our results indicate that DefogGAN can predict the enemy buildings and combat units as accurately as professional players do and achieves a superior performance among state-of-the-art defoggers.

Introduction

The success of AlphaGo (Silver et al. 2016) has brought a significant attention for artificial intelligence in games (game AI). Agents trained by deep reinforcement learning have demonstrated hands-down victories over expert human players in classic games such as Chess (Silver et al. 2018), Go (Silver et al. 2016), and Atari (Mnih et al. 2013). With more complex setting, real-time strategy (RTS) games serve a means to evaluate state-of-the-art learning algorithms. Game AI today opens up new opportunities and challenges for machine learning. The benefits of developing game AI are widespread beyond gaming applications. The exploration to adopt an intelligent agent in science (e.g., predicting protein folding in organic chemistry (Evans et al. 2018)) and enterprise business service (e.g., chatbots (Satu, Parvez, and Shamim-Al-Mamun 2015)) is making to enter a new era for game AI.

In this paper, we describe DefogGAN that takes a generative approach to compensate imperfect information presented to a gamer due to the fog of war. We use StarCraft, an RTS game featuring three well-balanced races for a gamer to choose and build substantially different playing styles and strategies. StarCraft remains a popular E-sport after more than two decades of the original release. In a daunting aim for our game AI to conquer highly-skilled human players, we train our DefogGAN with more than 30,000 episodes of expert and professional human replays. Such aim has been notoriously difficult for StarCraft whose long withstanding popularity has compounded a broad range of adept game tactics in addition to micro-control techniques (Ontanón et al. 2013) widespread in the E-sport scenes and `Battle.net`.

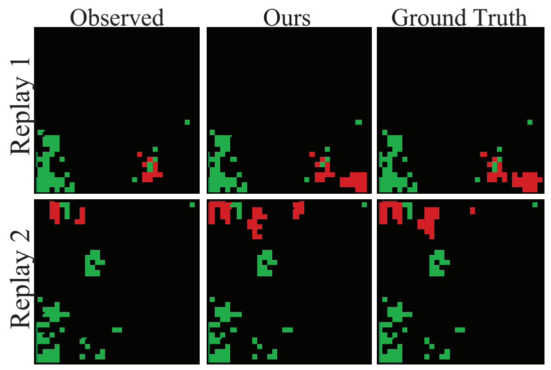


Figure 1: Comparison of DefogGAN prediction to ground truth. Friendly and enemy units are represented as green and red in the map (black). The unobserved enemy units are predicted by DefogGAN.

The fog of war refers to the lack of vision and information on an area without a friendly unit around it, including all regions that have been previously explored but left unattended currently. Partially Observable Markov Decision Process (POMDP) (Monahan 1982) best describes the fog of war problem. In general, POMDP gives a practical formulation for most real-world problems characterized by having many unobserved variables. For game AI, solving a partial observation problem is essential to improving its performance. In fact, many existing approaches to design intelligent game AI often suffer from the partial observation problem (Xu et al. 2018). Recently, generative models are used to alleviate the uncertainty of partial observations. The agent's performance is enhanced from taking advantage of the (predictive) results obtained through a generative model (Synnaeve et al. 2018; Kahng et al. 2018). The generative approach, however, cannot fully match highly skillful scouting techniques of a top-notch professional human player.

StarCraft provides a great platform to study complex POMDP problems related to game AI. We set up DefogGAN to accurately predict the state of an opponent hidden in the fog using the *realistic* information generated, thanks to generative adversarial nets (GANs) (Goodfellow et al. 2014). We find empirically that GANs generate more realistic images than variational autoencoders (VAEs) (Kingma and Welling 2013). To generate a defogged game state, we have modified the original GAN generator into an encoder-decoder network.

In principle, DefogGAN is a variant of conditional GAN (Mirza and Osindero 2014). Utilizing skip connections, the DefogGAN generator is trained on residual learning from the encoder-decoder structure. In addition to the GAN adversarial loss, we set up a reconstruction loss between fogged and defogged game states to emphasize the regression of unit positions and quantities. This paper makes the following contributions.

- We develop DefogGAN to resolve a fogged game state into useful information for winning. DefogGAN makes one of the earliest GAN-based approaches to cope with the StarCraft fog of war;
- Using skip connections for residual learning, we have set up DefogGAN to contain past information (sequence) in a feedforward manner without introducing any recurrent structure, making it suitable for real-time uses;
- We empirically validate DefogGAN in ablation study and other settings such as testing against extracted game intervals and the current state-of-the-art defog strategy.

Our dataset, source code, and pretrained networks are available online for public access.[1]

Related Work

StarCraft AI

StarCraft is an immensely successful RTS game developed by Blizzard Entertainment. Since its original release in 1998, StarCraft has attracted professional E-sport leagues and millions of amateur enthusiasts worldwide. Consisting of three fictional races, namely Terran, Protoss, and Zerg, StarCraft is considered as one of the most well-balanced online games ever created. The combinatorial complexity of player actions is extremely high, although at a high level, winning conditions for StarCraft can be built upon the military power and an economy accumulated by the player.

StarCraft AI has a long history, reflecting a number of different playing styles. Ontanon et al. (2013) point out that StarCraft playing essentially comprises two tasks. First, micro-management refers to the ability to control units individually. Good micro-management can keep a player's worker and combat units alive for a long time. Secondly, macro-management is the ability to produce units and expand the production facilities into regions other than the start location.

Defogging can be crucial to both micro- and macro-aspects of the game. Better estimation of hidden areas in the map will help win combats while the player has a higher chance of making the right decision for the future. A poor observation in general can hurt macro-management (Weber, Mateas, and Jhala 2011; Xu et al. 2018). Scouting is the most straightforward defogging technique (Park et al. 2012; Si, Pisan, and Tan 2014). Interestingly, Justesen & Risi (2017) propose a deep learning-based approach to learn the opponent status from units and upgrades information. Generative models give a new class of prediction techniques in StarCraft AI. The convolutional encoder-decoder (CED) model (Synnaeve et al. 2018; Kahng et al. 2018) can be used to recover information hidden in the fog. Synnaeve et al. (2018) find beneficial to use a convolutional encoder and a convolutional-LSTM encoder. Our approach of using GAN to generate hidden information as a predictive measure is new to the literature.

[1]https://github.com/TeamSAIDA/DefogGAN

Generative Adversarial Nets (GAN)

Goodfellow et al. (Goodfellow et al. 2014) introduce GAN to generate data from probabilistic sampling. GAN constitutes two neural nets, a generator G and a discriminator D, trained in the competition described by a minimax game:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{real}}}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

Radford, Metz, and Chintala (2015) have proposed DCGAN that uses a deep convolutional neural net as G. Vanilla GAN is trained on the Jensen-Shannon divergence (JSD), which can cause the vanishing gradient and mode collapse problems. WGAN (Arjovsky, Chintala, and Bottou 2017) proposes the use of the Wasserstein-1 metric to improve the vanilla GAN problems. Gulrajani et al. (2017) propose WGAN-GP having a gradient penalty that has a similar effect as the weight clipping. Zhao et al. (2016) introduce Energy-based GAN (EBGAN) using an autoencoder. Berthelot, Schumm, and Metz (2017) propose BEGAN that combines the WGAN and EBGAN ideas. We will experimentally compare the GAN variants for defogging performances.

Generative Approaches for Defogging

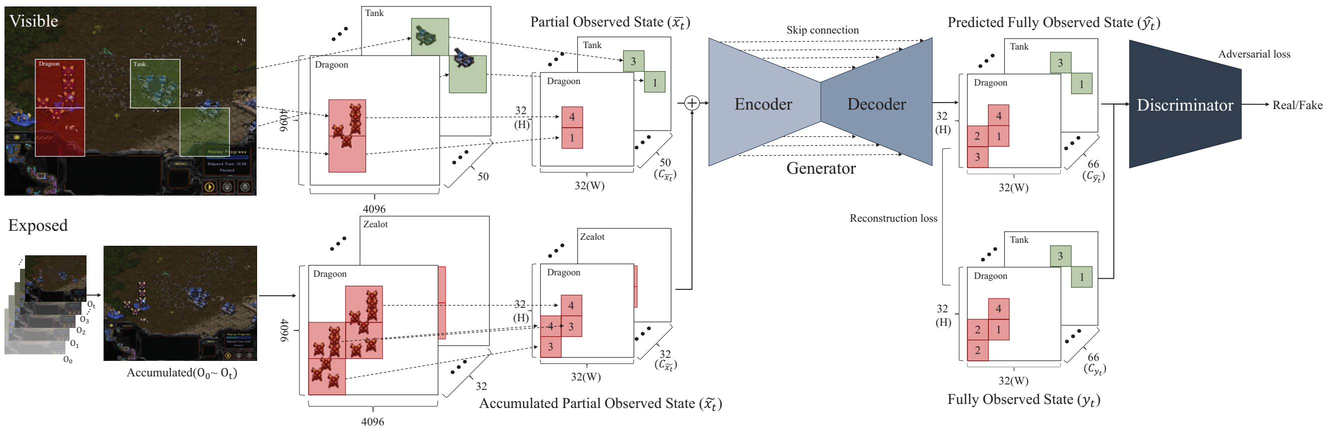
The fog of war problem is similar to inpainting (Nazeri et al. 2019) and denoising (Kingma and Welling 2013). However, there are three key differences. First, the enemy units may be hidden even in the presence of the friendly units, so defogging must predict the location and the number of each enemy unit type in a 2D grid space up to 4096×4096. Secondly, defogging is a regression problem, which must infer the number of units in the entire area based on a partial observation. Lastly, the problem is not just to generate an image based on the masked (fogged) image. Defogging must indicate the grid where a unit of interest is likely to exist.

DefogGAN

This section presents DefogGAN, explaining its architecture and objective functions. We also describe our implementation details.

Overview

DefogGAN generates a fully observed (defogged) state from a partially observed (fogged) state at a time t. For StarCraft, a fully observed state includes the exact locations



Visible
Exposed
Accumulated(O_0~ O_t)
Tank
Dragoon
Zealot
4096
50
32
Partial Observed State ($\bar{x}_t$)
Accumulated Partial Observed State ($\tilde{x}_t$)
32 (H)
32(W)
Skip connection
Encoder
Decoder
Generator
Reconstruction loss
Predicted Fully Observed State ($\hat{y}_t$)
Fully Observed State (y_t)
66
Discriminator
Adversarial loss
Real/Fake

Figure 2: Architectural overview of DefogGAN.

of all friendly and enemy units at a given time. Figure 2 presents DefogGAN. Feature maps computed on the current partially observed state input are sum-pooled. Feature maps on the past observations are accumulated and concatenated to the current before entering the generator. The reconstruction loss between the predicted and the actual fully observed states and the discriminator adversarial loss are used to train the generator.

Notation

We denote y_t a ground-truth fully observed state at time t, consisting of the exact locations of all units in the game. It is represented as a three-dimensional array of width, height, and channels. Each unit type makes up a channel, and the size of a raw game image in StarCraft is 4096×4096 pixels. With 66 unit types, a 1-vs-1 StarCraft game state is $4096 \times 4096 \times 66$. We use $\hat{y}_t$ for a predicted fully observed state. A partially observed state at a time t is $\bar{x}_t$. In StarCraft, friendly units are always visible, making the half of the channels in the input fully observed. Ignoring the enemy buildings, which are static units, a partially observed state is an array of size $4096 \times 4096 \times 50$. Here, 50 channels include 34 channels for friendly units and 16 channels for enemy combat units. Partially observed states accumulated until time t is denoted by $\tilde{x}_t$. Accumulated partial observations, however, include enemy buildings and exclude friendly units, which are already a part in the current partial observation. This results in an array of size $4096 \times 4096 \times 32$ for $\tilde{x}_t$. Combining $\bar{x}_t$ and $\tilde{x}_t$, a concatenated total input x_t is applied to DefogGAN.

Accumulating Partial Observations

Unlike vanilla GAN that generates an image x_{fake} from a latent variable, Defog needs to generate a defogged observation $\hat{y}_t$ given a partial observation $\bar{x}_t$. Defog has an autoencoder generator instead of a deconvolutional net.

$$f(\bar{x}_t) = \hat{y}_t = G(\bar{x}_t) = \text{Dec}(\text{Enc}(\bar{x}_t))$$

If a partially observed state $\bar{x}_t$ lacks temporal information about moving units, it would be insufficient to learn how to generate a fully observed state y_t. Accumulated partial observation $\tilde{x}_t$ facilitates such temporal information. Later in the paper, we show that using accumulated partial observation as an input increases precision and recall. DefogGAN takes in concatenated $\bar{x}_t$ and $\tilde{x}_t$:

$$x_t = \bar{x}_t \oplus \tilde{x}_t.$$

Note that we use downsampled x_t and y_t. Since the size of a raw state is too large, we reduce it to 32 x 32. More specifically, as shown in Figure 2, a partially observed state $\bar{x}_t$ is now an array of size ($32 \times 32 \times 66$). The downsampling allows DefogGAN to efficiently learn how to generate a fully observed state while preserving semantic information of a state (Synnaeve et al. 2018; Kahng et al. 2018).

For using temporal information, we could use a recurrent neural net. Using a recurrent neural net, however, comes with some disadvantages such as information dilution and gradient vanishing. Since StarCraft has a relatively long playing time, recurrent nets in general should take too many frames (e.g., to infer game states for a 10-minute duration, 14,400 frames are needed). Our DefogGAN approach has opted for stacking past partial observations onto the current (Mnih et al. 2013). By incorporating accumulated partial observation $\bar{x}_t$, we derive the adversarial objectives

$$\begin{aligned}\mathcal{L}_G^{'} &= \mathcal{L}_{adv} \\ &= \mathbb{E}_{\bar{x}_t \sim X_{par}, \tilde{x}_t \sim X_{acc}}[\log(1 - D(G(\bar{x}_t \oplus \tilde{x}_t)))] \end{aligned} \tag{1}$$

$$\begin{aligned}\mathcal{L}_\mathcal{D} = &- \mathbb{E}_{y \sim Y}[\log(D(y_t))] \\ &- \mathbb{E}_{\bar{x}_t \sim X_{par}, \tilde{x}_t \sim X_{acc}}[\log(1 - D(G(\bar{x}_t \oplus \tilde{x}_t)))] \end{aligned} \tag{2}$$

Pyramidal Reconstruction Loss

We train the generator $G(\bar{x}_t \oplus \tilde{x}_t)$ by minimizing the reconstruction loss between a generated state $\hat{y}_t$ and the ground truth y_t. To further enhance the generator, we introduce pyramidal reconstruction loss as a sum of the MSE between multiple levels of pooling having different sizes (H, W). Figure 3 illustrates pyramidal reconstruction loss.

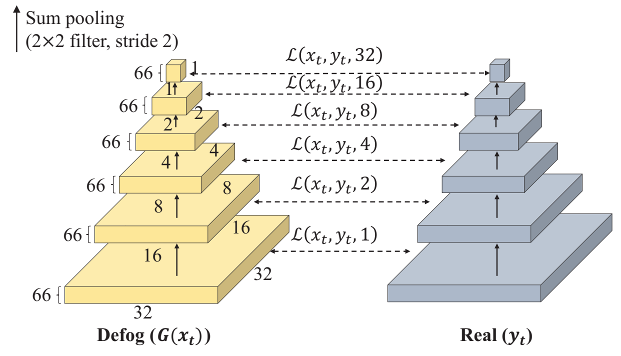

Sum pooling
(2×2 filter, stride 2)
$\mathcal{L}(x_t, y_t, 32)$
$\mathcal{L}(x_t, y_t, 16)$
$\mathcal{L}(x_t, y_t, 8)$
$\mathcal{L}(x_t, y_t, 4)$
$\mathcal{L}(x_t, y_t, 2)$
$\mathcal{L}(x_t, y_t, 1)$
66
1
2
4
8
16
32
Defog ($G(x_t)$)
Real (y_t)

Figure 3: Pyramidal reconstruction loss. The pyramid on the left shows the generation of $\hat{y}_t$ and the reduction of resolution through sum pooing. The pyramid on the right represents sum pooing of the ground truth y_t. The mean squared error (MSE) can be measured between the same resolutions. Note that pyramidal loss function $\mathcal{L}(x_t, y_t, s)$ measures the MSE with a stride s.

Multiple predictions at different scale are generated by sum pooling. By adjusting filter and stride sizes, sum pooling can generate multiple predictions in a pyramidal shape. More specifically, for a given feature map m_t, the sum pooling function sumpool(m_t, s) creates $m_t^{'}$ with a stride s and a filter size s. This can be formulated as follows

$$\text{sumpool}(m, s) = m_t^{'}(i, j) = \sum_{h=s\cdot j}^{s\cdot(j+1)-1} \sum_{w=s\cdot i}^{s\cdot(i+1)-1} m_t(w, h),$$

where (w, h) is the coordinates of m_t, and (i, j) is the coordinates of $m_t^{'}$.

At each generation, $\mathcal{L}_{dist}$ is evaluated as follows.

$$\mathcal{L}_{dist}(x_t, y_t, s) = \mathbb{E}\left[\|\text{sumpool}(y_t, s) - \text{sumpool}(G(x_t), s)\|_2^2\right]$$

For the resolution r (using $r = 32$) to be reduced, pyramidal reconstruction loss is evaluated by

$$\mathcal{L}_{rec} = \sum_{i=0}^{\log_2 r} w_i \cdot \mathcal{L}_{dist}(x_t, y_t, 2^i). \tag{3}$$

A scaling factor w_i adjusts the loss values at different scales. It is defined

$$w_i = \frac{4^{-i}}{\sum_{k=0}^{\log_2 r} 4^{-k}}.$$

The proposed pyramidal reconstruction loss allows DefogGAN to learn the total number of units with the lowest resolution of 1×1. Finally, by incorporating the reconstruction loss, the generator loss of DefogGAN is extended as

$$\mathcal{L}_G = \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec}.$$

Observation Preserving Connection

The DefogGAN encoder and decoder are connected in a symmetrical structure. We add residuals between the encoder and decoder at each layer to maintain the parts that have been seen already. By doing so, the generator learns the parts that are hidden in the fog (He et al. 2016; Isola et al. 2017). Through the encoder network, the compressed feature is well-communicated to the decoder for efficient learning. In particular, the observation preserving connections that tie the beginning and the end convey the information that has been observed already. This allows DefogGAN to focus on the information of the units that needs to be inferred. That is, the generator $F(x_t)$ learns by observing informational connection with $G(x_t)$ less the observed informational connection x_t:

$$F(x_t) = G(x_t) - x_t.$$

Total Objective

The total objective of DefogGAN is

$$\begin{aligned}\mathcal{L}_D = &- \mathbb{E}_{y\sim Y}[\log(D(y_t))] \\ &- \mathbb{E}_{\bar{x}_t\sim X_{par}, \tilde{x}_t\sim X_{acc}}[\log(1 - D(G(\bar{x}_t \oplus \tilde{x}_t)))], \\ \mathcal{L}_G = &\ \lambda_{adv}\mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec}.\end{aligned}$$

Note hyperparameters λ_{adv} and λ_{rec}. In this paper, we use $\lambda_{rec} = 0.999$ and $\lambda_{adv} = 0.001$.

Training

The overall training procedure of DefogGAN is presented in Algorithm 1. We use Adam (Kingma and Ba 2014) for training both discriminator and generator.

Algorithm 1 Training the DefogGAN model

```
$\theta_G, \theta_D \leftarrow$ initialize network parameters
$\lambda_{rec} = 0.999, \lambda_{adv} = 0.001, r = 32, epoch = 0$
repeat
    $X \leftarrow$ batch from dataset
    $\hat{Y} \leftarrow G(X)$
    $Y^{'} \leftarrow Y, \hat{Y}$
    $p_{data} \leftarrow D(Y^{'})$
    $p_g \leftarrow D(\hat{Y})$
    $\mathcal{L}_{rec} \leftarrow Eq.(3)$
    $\mathcal{L}_D \leftarrow Eq.(2)$
    $\mathcal{L}_{adv} \leftarrow Eq.(1)$
    $\mathcal{L}_G \leftarrow \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$
    //Update parameters according to gradients
    $\theta_G \leftarrow -\nabla_{\theta_G}\mathcal{L}_G$
    $\theta_D \leftarrow -\nabla_{\theta_D}\mathcal{L}_D$
    $epoch \leftarrow epoch + 1$
until $epoch = 1000$
```

Implementation

Generator The DefogGAN generator follows the style of the VGG network (Simonyan and Zisserman 2014). The filter size is fixed at 3×3. The number of filters doubles when the feature map size is reduced by half. DefogGAN does not use any spatial pooling or fully-connected layers but uses convolutional layers to preserve spatial information from input to output.

The DefogGAN generator consists of encoder, decoder, and a channel combination layer. The encoder uses $32 \times 32 \times 82$ input and extracts semantic features hidden in the fog by convolutional neural networks (CNNs). Each convolutional layer uses batch normalization and rectified linear unit (ReLU) to make the nonlinear conversion possible (Ioffe and Szegedy 2015; Nair and Hinton 2010).

The decoder generates predictive data using semantically extracted encoder features. The decoding process reconstructs data into a high dimension, and the inference is done using the transposed convolution operation. The decoder produces the same output shape as the input shape. We do not use as many convolutional layers as ResNet, considering the speed of learning due to the large initial channel size (He et al. 2016).

The final channel combination layer consists of a single convolutional layer, which combines the 82 channels of accumulated partial observations $C_{\tilde{x}_t}$ and $C_{\bar{x}_t}$ to obtain 66 channels $C_{\hat{y}_t}$ of information to predict. This infers $\hat{y}_t$.

Discriminator The DefogGAN discriminator is similar to that of DCGAN (Radford, Metz, and Chintala 2015). Three convolutional layers are used with a leaky ReLU activation function. Dropout is used for all layers instead of batch normalization. Through the fully connected final layer, predicting real or fake can be learned.

Experiments

Dataset

We have collected a large dataset of more than 33,000 replays of professional StarCraft players. Our experiments utilize replay log files, which contain detailed unit information. For each frame, a concatenated partial observation (x_t) of the fogged map $H \times W \times (C_{\bar{x}_t} \oplus C_{\tilde{x}_t})$ exists along the corresponding ground truth (y_t) in $H \times W \times C_{y_t}$. From each episode, we have decided to only use a portion from the 7th to the 17th minutes. This is because high-level units in a StarCraft game start to appear in about 7 minutes. Also, the game typically finishes in 10 to 20 minutes (Lin et al. 2017) although not many replays are of more than 17 minutes of duration. Our dataset comprises 496,830 frames. We use 80% of the data for training, 10% validation, and 10% testing.

Table 1 summarizes the DefogGAN input-output statistics, including partially observed states $\bar{x}_t$, accumulated partially observed states $\tilde{x}_t$, and ground truth y_t. On average, 54% of the total number of units are seen in partial observation, and 83% are seen in accumulated partial observation. Note that accumulated partial observation causes a type 1 error (i.e., false positive) because accumulated states contain the previous locations of moving units that are obsolete at the current time. Given this output space, the defog problem is to select an average of 141 spaces out of 67,584 ($32 \times 32 \times 66$) spaces possible.

Table 1: Confusion matrix of $\bar{x}_t$ and $\tilde{x}_t$. Using test data (more than 10,000 frames), average numbers are shown.

		y_t (GT)		Total
		Exist	Not exist	
$\bar{x}_t$ (partial)	Exist	81.58	0	81.58
	Not exist	59.35	67443.07	67502.42
Total		140.93	67443.07	67584.00

		y_t (GT)		Total
		Exist	Not exist	
$\tilde{x}_t$ (accum.)	Exist	109.49	7.30	116.79
	Not exist	31.45	67435.76	67467.21
Total		140.94	67443.06	67584.00

Evaluation Metrics

For performance evaluation, we compute five metrics:

Mean Squared Error (MSE) The MSE between $\hat{y}_t$ and y_t is

$$\mathrm{MSE} = \mathbb{E}\left[\|y_t - \hat{y}_t\|_2^2\right].$$

Our MSE criterion measures: 1) correct prediction of unit types present at each location 2) correct prediction of how many (if present).

Accuracy, Precision, Recall and F1 score Accuracy indicates how well the existence of units is predicted. Recall reflects how much false negative rate (type 2 error) is improved. For DefogGAN perspective, type 2 error gives a more practical indicator because the damage caused by an unexpected enemy (false negative) is greater than a nonexistent enemy (false positive). Precision represents a type 1 error as a percentage of what is expected to exist. The F1 score indicates the harmonic mean of recall and precision.

Determining Generator Training Interval

We have experimentally determined a reasonable amount of data needed for training the DefogGAN generator. Table 2 summarizes the generator performance measured in MSE, accuracy, and F1 score computed by varying number of frames used in training. Due to the nature of the DefogGAN prediction, the MSE criterion is most valuable. The empirical results suggest training with 10-sec worth of frames the best among our tested intervals.

Table 2: The DefogGAN generator performance comparison on varying number of frames used in training.

Interval(frame)	MSE	Acc.	F1	Recall	Preci.
5s(9,165)	0.00211	0.99942	0.854	0.808	0.906
10s(13,692)	**0.00208**	0.99944	0.856	0.807	0.913
30s(31,442)	0.00215	0.99945	0.860	0.808	**0.918**
60s(56,963)	0.00213	**0.99946**	**0.862**	0.814	0.915
600s(496,830)	0.00230	0.99944	0.859	**0.820**	0.901

Baseline

As presented in Table 3, accumulated partial observation $\tilde{x}_t$ results in better performance than partial observation $\bar{x}_t$.

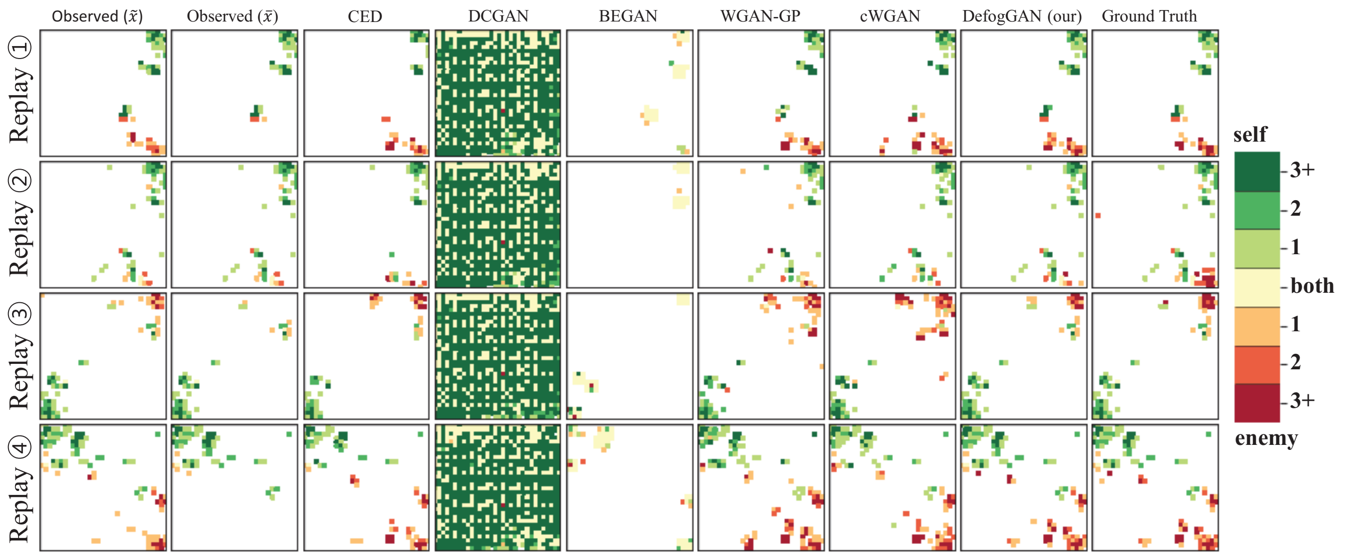


Figure 4: Visualization of prediction results. On far left, we show the accumulated partially observed states ($\tilde{x}_t$). The second column depicts a partially observed state, $\bar{x}_t$. The prediction result by CED, a state-of-the-art defogger is shown in the third column. Columns 4–7 are the results by DCGAN, BEGAN, WGAN-GP and cWGAN. Our DefogGAN result is presented in the eighth column, and the ground truth on the last. Rows represent replays used for evaluation.

Table 3: The overall accuracy performance of partially observed states $\bar{x}_t$ and accumulated partially observed states $\tilde{x}_t$. The MSE indicates how well the units are positioned and numbered in the unit map, and the Accuracy, F1 scores, Recall and Precision indicates how well the units are aligned in the unit map.

	MSE	Acc.	F1	Recall	Preci.
$\bar{x}_t$ (partial)	0.00548	0.99912	0.733	0.579	**1**
$\tilde{x}_t$ (accum.)	**0.00370**	**0.99943**	**0.850**	**0.777**	0.937

In fact, many rule-based agents leverage $\tilde{x}_t$ (Ontanón et al. 2013; Synnaeve et al. 2018). We take accumulated partial observation as our baseline.

Accuracy Comparison

In this section, we present a comparative performance analysis for DefogGAN. A rule-based StarCraft agent using accumulated partial observation is a reasonable baseline. This baseline means that a prediction model needs to make at least better prediction than just memorizing partial observation history. For comprehensive comparison, we select a diverse range of models including an autoencoder-based model CED (Synnaeve et al. 2018; Kahng et al. 2018), simple GAN-based models, DCGAN (Radford, Metz, and Chintala 2015) and BEGAN (2017), and WGAN-based models, WGAN-GP (2017) and cWGAN (Ebenezer, Das, and Mukhopadhyay 2019).

Comparison with baseline As shown in Table 4, DefogGAN results in a 44% decrease in MSE compared to the baseline. DefogGAN predicts the number of units in a given cell more accurately than the baseline. This is because DefogGAN is able to predict enemy units hidden in fog. On the other hand, DefogGAN seems to provide similar prediction performance in terms of accuracy and F1 score. Note that accuracy and F1 score do not measure how accurately the number of units are predicted, but just measure how accurately the existence is predicted. Then, the result can be understood that DefogGAN can predict the number of units much precisely while correctly predicting the overall distribution of units on a map.

Comparison with autoencoder model Compared to CED, one of autoencoder-based models, DefogGAN provide about 33% decreased MSE, and about 17% point increased F1 score. Note that recall of DefogGAN is very high, compared to that of CED. This high recall means that DefogGAN successfully discover enemy units hidden in fog. This high recall property is very important in StarCraft, since misdetected enemy units (i.e., low recall) can increase possible threat such as sudden attacks.

Comparison with GAN-based models DefogGAN makes a better prediction compared to other GAN-based models. As shown in Table 4, unconditional base GAN models such as DCGAN and BEGAN performs very poorly. This is mainly because these models are trained without reconstruction loss. WGAN-GP makes relatively good prediction results without reconstruction loss, but does not exceed DefogGAN. We carefully think that the Wasserstein distance of WGAN-GP makes an effect of reconstruction loss in training. Therefore, we do additional comparison with cWGAN, a WGAN variants that has reconstruction loss. However, cWGAN does not provide better performance than WGAN-GP.

Table 4: Accuracy comparison results. DefogGAN is compared with various other models.

	MSE	Acc.	F1	Recall	Preci.
Baseline	0.00370	0.99943	0.850	0.777	**0.937**
CED	0.00311	0.99896	0.682	0.538	0.933
DCGAN	2.16007	0.94844	0.019	0.239	0.010
BEGAN	0.01578	0.99353	0.024	0.039	0.018
WGAN-GP	0.00348	0.99885	0.701	0.648	0.763
cWGAN	0.00372	0.99878	0.688	0.644	0.737
DefogGAN	**0.00208**	**0.99944**	**0.856**	**0.807**	0.913

Visualization of prediction results The prediction performance of DefogGAN can be effectively explained with the visualization in Figure 4. We randomly select four replays and present the defogged fully observed states predicted by each model. For example, in replay 4, we cannot see red enemy units in the lower right corner of the partially observed state $\bar{x}_t$. Also, we can only see a subset of enemy units from the accumulated partially observed states $\tilde{x}_t$. By using both observation and accumulated observation, DefogGAN generates a fully observed state $\hat{y}_t$ that looks most similar to the ground truth. Since DCGAN and BEGAN do not use reconstruction loss, they fail to generate a fully observed state that has similar pattern to the ground truth. CED generates fairly plausible full states, but DefogGAN generates more accurate results. WGAN-GP generates plausible full states without reconstruction loss. However, it seems to have a tendency to generate false positive results (i.e., low precision). cWGAN (a WGAN-GP variant that additionally use reconstruction loss) seems to reduce such false positives, but still do not make a prediction better than DefogGAN.

Ablation Study

We evaluate the performance of the proposed method that combines accumulated partial observation $\tilde{x}_t$ and partial observation $\bar{x}_t$, joint loss and reduced resolution loss. Finally, we compare the performance of the observation preserving connection.

Table 5: Ablation study results. Each component is excluded from DefogGAN in order: current partial observation, accumulated past partial observation, adversarial loss, reconstruction loss, pyramidal loss (L_2 loss was used instead), and observation preserving connection.

	MSE.	Acc	F1	Recall	Preci.
$\ominus X_{par}$	0.00293	0.99930	0.831	**0.826**	0.836
$\ominus X_{acc}$	0.00426	0.99897	0.732	0.674	0.802
$\ominus L_{adv}$	0.00310	0.99887	0.662	0.529	0.882
$\ominus L_{rec}$	1.73986	0.97942	0.026	0.133	0.015
$\ominus$ Pyramidal	0.00210	0.99943	0.855	0.809	0.907
$\ominus$ Ob-conn	0.00401	0.99829	0.516	0.437	0.631
DefogGAN	**0.00208**	**0.99944**	**0.856**	0.807	**0.913**

DefogGAN proposed in Table 5 shows that our proposed techniques in the ablation study produce good performance.

Effect of concatenated partial observation Using the concatenated partial observation method, the MSE is 29% better than using only the accumulated partial observed information and 51% better than using only the partial observed information. This indicates that it is important to utilize past information. In addition, when used in combination with partially observed and accumulated partially observed information, the total number of units observed from the past is identified, and certain information without type 1 errors is used for learning. In other words, it contributes to the performance improvement by showing the number of units as much as possible and the units that can be confirmed as correct.

Effect of adversarial learning The third row of Table 5 shows the overall accuracy performance of DefogGAN when trained without adversarial loss. Without adversarial loss, the overall accuracy performance significantly decreases. MSE increases about 49% (i.e., from 0.00208 to 0.00310). F1 score decreases by 0.194 (i.e., from 0.856 to 0.662). In the area of image generation, learning with adversarial loss generates clearer images than learning with MSE loss (Pathak et al. 2016; Isola et al. 2017). In DefogGAN, we see a similar effect. We conjecture that adversarial loss also helps accurately predict the fully observed states of a game.

Effect of reconstruction loss Pyramidal reconstruction loss helps to learn fully observed states. Since it measures the difference between a predicted fully observed state and the ground truth at multiple scales, it helps DefogGAN accurately predict the total number of units hidden in the fog.

Effect of observation preserving connection As shown in the 6th row of Table 5, when trained without observation preserving connection, the overall accuracy performance of DefogGAN significantly decreases. More specifically, MSE increases about 200% (i.e., from 0.00208 to 0.00410). F1 score decreases by 0.340 (i.e., from 0.856 to 0.516). This can be considered as a similar effect that skip connection of U-Net (Ronneberger, Fischer, and Brox 2015) provides better results by allowing information to flow from input to output.

Conclusion

We have presented DefogGAN, a generative approach for game AI to predict crucial state information unavailable due to the fog of war. DefogGAN accurately generates defogged images of a game that can be used to improve win rates against expert human players. In our experiments with StarCraft, we have validated that DefogGAN achieves a superior performance against state-of-the-art defoggers. Improving on imperfect information during an RTS game play could bring substantially better macro-management overall, although this is an ongoing research area for game AI. DefogGAN is one of the earliest applications for adversarial learning to improve the fog of war problem, and it can be applied to other real-world POMDP problems.

Acknowledgment

The authors would like to thank Dr. Wonpyo Hong, CEO and President of Samsung SDS, whose vision in AI has led to create SAIDA Lab for advanced AI research.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Berthelot, D.; Schumm, T.; and Metz, L. 2017. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

Ebenezer, J. P.; Das, B.; and Mukhopadhyay, S. 2019. Single image haze removal using conditional wasserstein generative adversarial networks. *arXiv preprint arXiv:1903.00395*.

Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Zidek, A.; Nelson, A.; Bridgland, A.; Penedones, H.; et al. 2018. De novo structure prediction with deeplearning based scoring. *Annu Rev Biochem*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *NeurIPS*.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of wasserstein gans. *NeurIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*, 770–778.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proc. of CVPR*, 1125–1134.

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

Kahng, H.; Jeong, Y.; Cho, Y. S.; Ahn, G.; Park, Y. J.; Jo, U.; Lee, H.; Do, H.; Lee, J.; Choi, H.; et al. 2018. Clear the fog: Combat value assessment in incomplete information games with convolutional encoder-decoders. *arXiv preprint arXiv:1811.12627*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lin, Z.; Gehring, J.; Khalidov, V.; and Synnaeve, G. 2017. Stardata: A starcraft ai research dataset. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *NeurIPS Deep Learning Workshop*.

Monahan, G. E. 1982. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science* 28(1):1–16.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. of ICML*.

Nazeri, K.; Ng, E.; Joseph, T.; Qureshi, F.; and Ebrahimi, M. 2019. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*.

Ontanón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2013. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Trans. on Computational Intelligence and AI in games*.

Park, H.; Cho, H.-C.; Lee, K.; and Kim, K.-J. 2012. Prediction of early stage opponents strategy for starcraft ai using scouting and machine learning. *Proc. of the Workshop at SIGGRAPH Asia*.

Pathak, D.; Krahenbuhl, P.; Donahue, J.; Darrell, T.; and Efros, A. A. 2016. Context encoders: Feature learning by inpainting. In *Proc. of CVPR*, 2536–2544.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Satu, M. S.; Parvez, M. H.; and Shamim-Al-Mamun. 2015. Review of integrated applications with aiml based chatbot. In *2015 International Conference on Computer and Information Engineering (ICCIE)*.

Si, C.; Pisan, Y.; and Tan, C. T. 2014. A scouting strategy for real-time strategy games. In *Proc. of the 2014 Conference on Interactive Entertainment*, 1–8. ACM.

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. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Synnaeve, G.; Lin, Z.; Gehring, J.; Gant, D.; Mella, V.; Khalidov, V.; Carion, N.; and Usunier, N. 2018. Forward modeling for partial observation strategy games - a starcraft defogger. *NeurIPS*.

Weber, B. G.; Mateas, M.; and Jhala, A. 2011. A particle model for state estimation in real-time strategy games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*.

Xu, S.; Kuang, H.; Zhuang, Z.; Hu, R.; Liu, Y.; and Sun, H. 2018. Macro action selection with deep reinforcement learning in starcraft. *arXiv preprint arXiv:1812.00336*.

Zhao, J.; Mathieu, M.; and LeCun, Y. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*.