29th International Command and Control Research and Technology Symposium, 24th-26th London, UK

Topic 5: Non-Human 'Intelligent' Collaboration and Autonomous Systems

AI techniques in Gaming for C2. Strategy detection, explainability and usability in StarCraft II

Carolina Sanchez Hernandez
Cambridge Consultants
Carolina.sanchez@cambridgeconsultants.com

Sam Connolly
Cambridge Consultants
Sam.connolly@cambridgeconsultants.com

Matthew J. Clayton
Cambridge Consultants
Matthew.j.clayton@cambridgeconsultants.com

Michelle Lim
Cambridge Consultants
Michelle.lim@cambridgeconsultants.com

Emma Hughson
Cambridge Consultants
Emma.hughson@cambridgeconsultants.com

John Smeaton
Cambridge Consultants
John.smeaton@cambridgeconsultants.com

Dan Valentine
Dan.valentine@cambridgeconsultants.com

Aravind Pravhakaran
Aravind. Pravhakaran
@cambridgeconsultants.com

Joe Smallman
Joe.smallman@cambridgeconsultants.com

Abstract

Video games are an ideal type of simulation environment to create and test new AI developments, as they provide a controllable set of variables and baselines to test results against, and data availability to explore, therefore offering a rapid turnaround on finding which technique is most valuable and has most potential for real-world applications.

As part of the Machine Speed C2 (MSC2) programme run by the UK Defense, Science and Technology laboratory (Dstl), our project explores: (i) piercing the 'fog of war' to infer opponents' positions and (ii) investigating key elements for detecting opponent's strategy within the game StarCraft II. As with real-world C2, this is a complex game in which a player must make decisions with partial information available in order to defeat an opponent. Our goal consisted of developing two AI assistants to augment a human player's decision making by: 1) developing an encoder-decoder neural network architecture to enable nowcasting likely locations of enemy units through the fog of war with an associated confidence level 2) using unsupervised techniques (clustering) and supervised techniques (a neural network classifier) to classify strategies with confidence and probabilistic metrics 3) creating a user-centric explainability framework that translates technical outputs into visual and text explanations based on user-centric understanding 4) performing a usability study to determine which method of communicating the AI outputs is more valuable to non-expert users.

Details and conclusions of this study are presented with recommendations as to how to use the learnings of this project within a C2 environment.

Keywords: fog of war, strategy detection, explainability

1 Background

Within this paper, we demonstrate the value of artificial intelligence (AI) research for military C2 using the commercial video game StarCraft II (SC2). We have focused our research on how we can use gaming as a type of simulation environment to progress development of AI techniques, and how this might help enable further research within C2. Our approach to this project has been to use the gaming environment not to create yet another AI super player, but to create AI agents that can assist and augment the human player's insights into opponents' locations and strategies, to inform the player's decision making.

The importance of the gaming industry for the development and study of AI algorithms is undeniable. Large proportions of video games involve strategic decision making, optimization processes or competition, which make up fertile exploration grounds for AI, and many games accumulate the large amounts of data required for advance machine learning techniques such as deep learning. For these reasons, AI researchers are making significant use of video games as training environments and benchmarks. These games have a wide range of features and attributes that create challenged for AI research, spawning many new algorithms in the last decade.

A significant amount of AI research has made use of the game (SC2), a science-fiction-themed real-time strategy (RTS) video game. Released in 2010, SC2 is one of the most popular RTS games ever made, with over 6 million copies sold. It is a popular e-sports game, typically played in 1v1 matches online.

The game's developer, Blizzard Entertainment, have made available an application programming interface (API) for extracting data from the game and enabling AI agents to interact with it, along with a large dataset of recorded games played by human players on the game's competitive ladder. This has facilitated a huge amount of research and AI development.

1.1 Aims of the research

There are significant challenges facing the development and transformation of C2 operations going forwards, from a technical perspective, a human perspective, and interactions between humans and AI agents.

The overall environment within which the future C2 concepts are being built will constitute a HAC (human/agent collective), which could take many forms. For example, AI agents could perform tasks autonomously, the outputs of which could then be used by human to improve their analysis or decision making; humans performing tasks could work together with AI agents to deliver key outcomes; and there are tasks and roles that might include inputs/outputs and collaborations between humans and AI agents.

This undertaking is ambitious. Video games can play an important part by providing a simulation testbed for some of these elements within different future C2 concepts. Games resemble formalized, if simplified, models of reality, and by solving problems in these environments we can learn how to solve analogous problems in reality.

In this study we focused on three main areas of research:

1.1.1 Exploring the challenge of user-centric AI explainability for human augmentation of decision making

The inability to explain or to fully understand the reasons why AI algorithms perform as they do is a real problem for practical implementation and trust. The gap between the research community and business sectors has been impeding the full penetration of the newest ML models in sectors that have traditionally lagged behind in the digital transformation of their processes [6]. This is even more important in military contexts in which it will be essential for C2 users in operations and warfare to understand, appropriately trust, and effectively manage an emerging generation of artificially intelligent machine partners. For example, an intelligence analyst who receives recommendations from a bigdata analytics system needs to understand why it recommended certain activity for further investigation. Similarly, an operator who tasks an autonomous system needs to understand the system's decision-making model to appropriately use it in future missions.

In section 2, we describe how we addressed these issues.

1.1.2 Exploring the challenge of partial information to infer opponents' positions.

To design and execute operational procedures, a military expert needs specific operational information, which influences (positively or negatively) all phases of a mission. One common challenge when gathering information about a mission is information gaps, due to missing components of operational information which can be of crucial importance, preventing experts from making correct assessments. Another important challenge is dynamic information, with factors rapidly changing over time and information becoming out of date, and therefore constantly needing to be updated [1].

In classic military terms, the 'fog of war' refers to the diminished level of accuracy and reliability of information exchanged in times of war, and the difficulties encountered by political and military leaders when seeking to compensate for this limitation and maximize the value of the data used for taking decisions [2].

Methods by which fog of war are typically included in real-time strategy video games such as SC2 include the obscuration of information relating to areas which are unexplored or not within sight range of the player's base and troops (typically including the area around the enemy base), enemy assets, troop locations and unknown terrain. Fog of war mechanics make only limited portions of the map viewable. Unit movement shifts these viewable zones and causes previously visited areas to fade out of sight. Progression requires an eventual confrontation with whatever lies in the surrounding fog, forcing players to think strategically about preparing for these unknowns [3].

Within our project, we investigated how to use information on the full current state of the game based on historical data and the current known portion of the game state, to create an AI agent that nowcasts a prediction of the full game state, indicating what most likely lies behind the fog of war. To assess these outputs, we also developed a methodology to estimate confidence levels on these outputs. These aspects of the project are described in Section 3.

1.1.3 Exploring the challenge of extracting key features for opponent strategy detection

Strategic thinking is greatly concerned with the consideration of one's own ends, ways, and means. This is a necessary component of strategic analysis, and it can only be achieved if ways, and means are considered with relation to the enemy, as strategy is necessarily adversarial [4]. Different enemies present different challenges, different ends, different rationalities to conceive of these ends, and differing levels of commitment to these ends; different strategic ways in which they can operate, and different strategic cultures.

Related to the above, opponent-modeling research uses data gathered from past experience, or even online, to complete or refine models of opponent behavior. Observation of the enemy's ways can offer the opportunity to learn about the enemy's assumptions [4].

Recent advances in tracking technology and both supervised and unsupervised machine learning, coupled with the clear need to move beyond the present limitations of model-based approaches, have given rise to a growing number of techniques for opponent modeling [5].

In our research, we have used both unsupervised and supervised machine learning to characterize and classify opponent strategies within a SC2 game in real time and changes of strategy over the timeframe of the game, based on streamlined key features that help to identify those strategies. These aspects of the project are described in section 4.

2 User-centric approach to augment the user decision making: a user-centric explainability framework

As described in [9], it is becoming apparent within the research and innovation community that:

- AI Explainability techniques (XAIs) have been developed by experts for experts. There is a disconnect between technical XAI approaches and supporting users' end goals in usage contexts [7] and therefore, there is a need for new approaches.
- Interpreting the outputs of XAIs for different users is extremely important. There is a disconnect between assumptions underlying technical approaches to XAI and people's cognitive processes [7]. The user perspective is needed to provide enough information for understanding of and trust in AI tools.
- Both the development of more user-focused XAIs and their interpretability for different uses and tasks is crucial for human-machine teaming to be successful. Potential inequalities of experience and understanding can lead to mistrust and misuse of AI [7].

New user-centric XAIs are starting to being explored such as knowledge graphs and neuro-symbolic XAIs [10], but there is a need to bridge the gap between current practices and user understanding.

In this project, we created a framework that aims to bridge this current gap within our gaming task, whilst also pointing to future links with XAI development, social sciences and behavioral sciences, in order to create the right socio-technical solutions.

2.1 User-centric Explainability framework

Our user-centric approach aims to bridge the gap between current "expert-focused" approaches to AI development and AI explainability techniques (XAIs), and actual user needs, in order to allow them to understand the AI outputs and incorporate them into their decision making (Figure1).

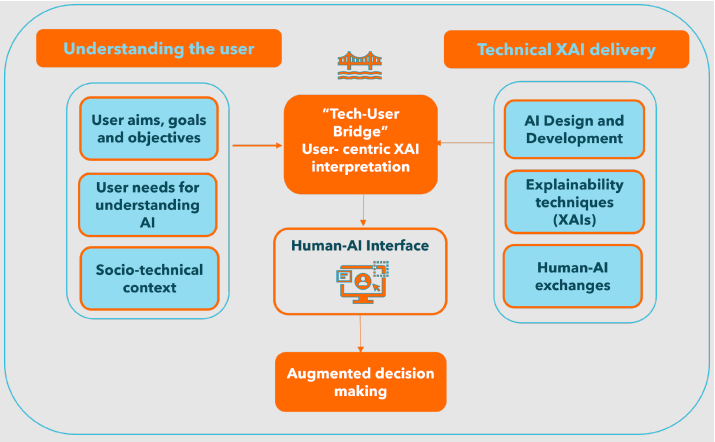

Figure 1 Summarised User-centric explainability framework

User understanding comes from different elements. Understanding the user goals and objectives, identifying where AI could add value, and their needs for using and understanding this tool should be at the centre of any consideration for AI development. The design and development of the AI tool needs to meet the requirements of the context of the application and tasks that the person will perform, but also how this tool will interact with the user (whether this is for assisting the user with information, teaming, cooperating, or performing an autonomous task that needs to be overseen).

It is therefore essential to understand the user needs in terms of the information needed to interpret the AI outputs and use them with confidence and to their maximum value. In many cases, the user might not have any technical knowledge of AI tools, but still needs relevant and targeted information to decide on how to use that AI tool for decision making. Mental models are one of the approaches that have been explored, to make sense of the perceptions and beliefs of the user before exposing them to an AI tool, so that this tool can be designed to address and enhance the mental model of the user regarding the expectations of the AI and its use. We explored this approach during our project; however we did not implement it due to time constraints. However, we believe that this is one of the key approaches to explore going forwards to define what we mean by user needs and how these might change depending on initial beliefs.

There is also a need to acknowledge that humans and AI interactions do not occur in isolation. The implementation of AI within different contexts implies the consideration of a socio-technical system where several users will interact with several AI tools and agents, explored within the MSC2 programme as the Human Agent Collective (HAC) for C2.

Related to human-AI understanding, a huge amount of research is happening on the subject of AI explainability techniques (XAIs). However, the outputs of these techniques are normally very "expert" like, which require a huge amount of technical understanding of AI to be able to make sense of them.

Human-AI interactions are a key element of research and development to deliver AI solutions, but are also thought out in a very mechanistic way and not very flexible or adaptable to different users within a HAC.

To bridge this gap and deliver the right information in the right format for the user, within our project we considered:

1) Who the users of the AI tool were:

- We used a player of SC2 as a proxy for a decision maker for C2. The player needs to assess the situation of the game with limited information and hypothesize opponents' locations and movements to make their own decisions on: (i) prioritisation of own resources for intelligence, surveillance, and reconnaissance (ISR) (ii) decision making on strategy and actions that can give them advantage in the game.

2) What the user requirements for AI functionality were:

- We assessed two functionality requirements within the game to address the user needs: (i) piercing the fog of war to infer opponents' locations and (ii) classifying opponents' strategy and changes in strategy over time. These two AI assistants give the player enough information to augment their decision-making capabilities.

3) What the user requirements for AI explanations were:

- We assumed the human (player) did not have technical knowledge of AI and therefore consider needs for understanding of AI and AI outputs for decision making.

4) The socio-technical context within which the AI tool would be used was:

- We took the gaming environment as a learning platform for human-AI interaction. We explored through a usability study how humans (players) would interact and use the information provided. We considered how to explore this further and learnings for C2 decision making.

The following sections will explain in detail how we addressed the above user needs.

3 AI for inferring opponents' positions from partial information

3.1 Previous work

SC2 has been a popular platform for researchers to develop and demonstrate cutting edge AI systems. One of the most important AI developments leveraging SC2 is DeepMind's AlphaStar [11]: a model trained using Reinforcement Learning to play SC2 to a level equivalent to top human professional players.

Our work on hidden state estimation in SC2 builds on the existing body of work in this area using SC2 as well as its predecessor game StarCraft: Brood War. We leverage the SC2API software developed by DeepMind, and apply techniques inspired by work performed by teams at Meta [12][13], and Samsung [14], who developed AI approaches for estimating aspects of the game state hidden by the fog of war using a model we call a "defogger", following [13].

3.2 Data sets and data extraction

The dataset used in this work consists of 79,806 recorded games of SC2 ("replays") played on the open ladder by human users on game versions 4.9.3 and 4.10.0. This data was sourced from Blizzard's Game Data API [15].

Replay files do not contain full state data themselves, instead containing only basic metadata and the series of clicks made by the players to take actions in the game. State data was therefore extracted by using the SC2 API to run the replays with the SC2 game engine and save the full state for a given game frame. Each replay contains all data required to completely reconstruct the game from either player's perspective. As the SC2 game engine is deterministic, re-running this extraction returns the same extracted data every time.

As the SC2 ladder contains a wide variety of player skill level, we limit the data set to games between players with a minimum MMR (Matchmaking Rating, a measure of player skill) of 1000 and minimum actions per minute (APM) of 10; this eliminates a small fraction of players who perform poorly. The dataset therefore consisted only of games involving experienced and expert players, who make up the majority of the SC2 ladder population.

The dataset was also limited to games in which both players used the 'Terran' race – one of 3 races available in SC2. The Terran race is a human or human-like race with military units and structures which are more similar to real-world military assets than those of the other two (alien) races, albeit still having Science Fiction aspects due to the game's setting. For this reason, games including the other two races (Zerg and Protoss) were excluded.

Every 45th game frame was extracted – as the game typically runs at 22.5 frames per second, this lead to an extracted data rate of 1 frame every 2 seconds. The dataset contains games the majority of which are 5 minutes to 60 minutes long, with a typical game lasting around 15 minutes, i.e., around 450 extracted frames. The first 30s of each replay were discarded, as this period is essentially the same for all games on a given map.

The dataset was split randomly into training, validation and test sets of full games, in the ratio of 80:10:10. The training split (63,844 games) was used to optimize the neural network parameters during training. The validation split (7,979 games) was used to measure performance during training. This measurement was in turn used to optimize network hyperparameters such as learning rate and model architecture. The test split (7,983 games) was used to compare model performance on data that was not used to optimize the network parameters or hyperparameters. Quantitative results shown later in this paper are calculated using this data split, to provide unbiased estimates of performance. For both training and evaluation, batches of sequences were randomly drawn from all periods of the game.

3.3 Model architecture

For our 'defogger' model, we use a neural network architecture with an encoder-decoder design inspired by a the SC2 defogger model used in [13], and similar to the popular U-Net structure [16]. The model works similarly to an autoencoder: encoding the input into a lower-dimensional latent space – a model-defined projection space in which datapoints appear as vectors – then decoding it into the output space in which datapoints have our chosen output format. We add a recurrent temporal core between the encoder and the decoder, to allow the model to process multiple input frames to generate an embedding of both temporal and spatial information, as well as an auxiliary output head to generate non-spatial outputs and stabilize training. In

addition, following U-Net, we make use of 4 skip connections to allow spatial information to pass directly from the encoder to the decoder without passing through the core, making it easier for the model to utilize spatial information in the input. The model was implemented using the deep learning framework PyTorch [17]. A diagram of the overall architecture is shown in Fig. 2.

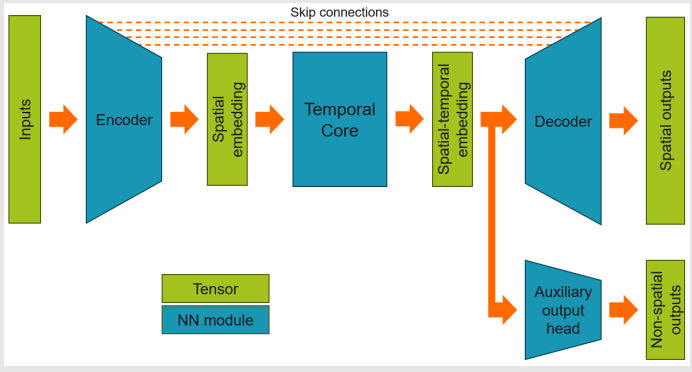

Figure 2. Diagram of the key components of the neural network architecture used in the 'defogger' model for predicting the full game state from the partial state.

Model inputs

The inputs to the 'defogger' model consisted of 87 spatial data channels consisting of 64 x 64 grid maps containing different information from the partial game state extracted from a single frame of an SC2 game. The data in each of the 87 channels was assigned as follows:

- 1-40: Player unit counts for each of the 40 Terran unit types (structures, vehicles and infantry), showing the number of friendly units of each type, present in each cell.
- 41-80: (Partial) opponent unit counts for each of the 40 Terran unit types, showing the number of units of each type present in each cell. Only the counts for units that are visible to the player or for previously seen structures (in their last known locations) are included.
- 81: 'Blip' counts for unclassified opponent units observed by a 'Sensor Tower' (essentially "radar blips"). This channel counts the number of blips in each grid cell.
- 82-83: Resource levels for Vespene gas and Minerals (the two resources which the player can extract from the terrain) at each grid cell. This input only provides information on whether these resources are present on the map, not whether they are being extracted. All resource deposits are visible to the player from the start of the game, however if the opponent has extracted resources from a deposit not visible to the player, the input will still show the last-seen quantity of remaining resources (or the full quantity from the start of the game if it has not yet been observed at all).
- 84: Map visibility: 1 if that grid cell is visible to friendly units, 0 if not visible. When part of a grid cell is visible, this value shows the fraction of the cell which is visible.
- 85: Traversable regions of map: 1 if units are allowed to "walk" on that grid cell, 0 if the cell cannot be walked on (e.g., if it is part of a cliff or other obstacle). When part of a grid cell can be walked on, this value shows the fraction of the grid cell that can be walked on. This restriction only affects ground units: air units can traverse all grid cells.
- 86: Map regions that can be built on: 1 if buildings can be constructed in that cell, 0 if they cannot. When part of a grid cell can be built on, this value shows the fraction of the cell that can be built on.
- 87: Current game time as a count of how many game frames have elapsed since the start of the game (there are 22.5 of these per second of game time). As this is a single, global, non-spatial parameter, its value is the same in each grid cell, so the same value is repeated at every point in the 64x64 grid. The game time is input in this way to allow all inputs to be the same shape, and to allow the game time to be used by all cells in subsequent convolutions.

Model outputs

Similar to the model input, the model outputs the spatial distribution of 81 output channels on a 64x64 grid. These channels are not the same as the input channels, and contain the following:

- 1-40: Predicted player unit counts per grid cell for the 40 Terran unit types for each cell (these should match the input channels 1-40, as this information is fully available to the model in the input).
- 41-81: Full estimated opponent unit counts per grid cell for the 40 Terran unit types for each cell – this should accurately reproduce visible opponent units in the input, as well as predicted unit placements behind the fog of war.

- 81: Estimated likelihood that mineral mining is occurring at a Command Center/Orbital Command/Planetary Fortress in that cell (including locations behind the fog of war). This estimates the locations of enemy bases performing resource extraction.

Additionally, the auxiliary output head produces a length-40 vector of non-spatial total unit counts for each opponent unit type across the entire map.

Encoder

The encoder is made up of a bank of 2D Convolution layers followed by a Linear layer, which compresses the 87 64 x 64 spatial input channels into a set vectors of length 512 for each input frame. These vectors comprise global state embeddings which encode all the information derived from the map for that frame.

Sigmoid linear unit (SiLU) activations introduce non-linearity after each layer. We apply Batch Normalization after each layer to improve training performance. Input sequences are processed as batch samples until the final layer, whereupon the tensor is reshaped to include a sequence dimension for input into the temporal core. Skip connections leave the encoder at each convolutional layer, to allow spatial information to "leak" to the decoder directly, which helps reconstruct spatial detail in the output map.

Temporal core

The temporal core was incorporated to provide a mechanism for the model to accumulate information from multiple input frames and provides it with "memory". We use an LSTM (Long Short-Term Memory [18], as it is a standard approach for processing sequence data.

Decoder

The decoder transforms the spatiotemporal embedding output by the temporal core back into the spatial data which make up the output map, effectively doing the reverse of the encoder. This is done by reshaping this output latent vector into a spatial representation with a low resolution (8x8), then passing it through a series of transpose convolution layers to expand the dimensionality of the latent representation tensor up to the 64x64 output resolution.

Prior to each transpose convolution layer, the intermediate latent representation tensor is combined with data from the skip connections originating in the encoder at each resolution increment. These tensors are combined by concatenating them along the channel dimension and then using a convolution layer with a 1x1 kernel to combine the two representations.

We apply a Rectified Linear Unit (ReLU) nonlinearity after every convolution and transpose convolution layer, and we apply Batch Normalization after every transpose convolution layer.

We apply a final sigmoid activation to the mining likelihood output to force it to the range [0, 1], as it represents a probability.

Auxiliary output head

The auxiliary output head takes the spatiotemporal embedding vector produced by the temporal core and uses it to estimate the total number of each type of unit present in the entire map, without regard to where on the map those units are. This head performs two functions:

- The global unit count estimates produced by this head are used to calculate an auxiliary loss which improves the training stability of the network.
- The dedicated global unit count estimates provided by this output may have different error properties to a global estimate generated by summing across grid cells in the main spatial output. This is useful as a diagnostic during training to monitor the status of the temporal core.

The auxiliary head is made up of a set of linear layers, as it deals with the linear global vectors rather than spatial maps, along with Batch Normalization blocks and SiLU nonlinearities.

3.4 Defogger training approach

Loss function

The loss function used to train the model has two components:

- A component calculated from the high-dimensional (spatial) model output generated by the decoder. This loss provides the primary training signal for the model's predictions by comparing the decoder's output to the true full game state. This loss component in turn consists of two subcomponents:
  - A Mean Squared Error (MSE)

unit-count-prediction loss applied to the 80 output channels that predict unit counts.

- A mining-prediction loss applied to the single output channel that predicts the probability of the opponent mining in each grid cell. As this is a probabilistic output, a Negative Log Likelihood (NLL) loss is used.

- A component calculated from the auxiliary output head which splits off after the temporal core. This loss stabilizes the training of the encoder and temporal core, and ensures that the model generates a rich embedding in the temporal core, rather than relying only on the skip connections which bypass the core (and the auxiliary output head). An MSE unit-prediction loss is used here.

The components of the loss function are combined by addition, and are balanced using scaling factors α_n. The total loss is therefore given by:

$$Loss = \|y_{unit} - \hat{y}_{unit}\|_2^2 - \alpha_0 \left(\sum_{grid\ cells} y_{mining} \log log \left(\hat{y}_m \right. \right.$$

Where y is the true game state and $\hat{y}$ is the game state estimated by the model. During training, loss components were tracked individually (in addition to the total loss) to allow performance of different aspects of the model's predictions to be monitored.

Learning rate schedule

Our training process utilised a 1-cycle cosine annealing learning rate schedule which exploits the phenomenon of super convergence [19], following the recommendations [20]. This schedule performs an initial warm-up phase at a low learning rate, then cosine-anneals the learning rate to a value significantly above the optimal value to enable super-convergence, before finally cosine-annealing it to zero to overcome stochastic noise.

This approach enabled a reduction in training time by a factor of 5 when compared to traditional deep learning schedules which use a constant learning rate, or one that anneals downwards during training.

For our model, the best-performing constant learning rate value is 0.001, which requires approximately 1000 epochs of training to reach good performance. However, the 1-cycle approach required only 200 epochs to surpass the constant-learning-rate performance. A secondary advantage of this approach is reducing overfitting due to the increased gradient noise at the higher peak learning rate.

Training process

The defogger model was developed, and its performance improved, by devising a set of experiments which tested the effect of variations in model architecture and training approach, with the results of each experiment informing later experiments. In total, we performed approximately 600 training runs to optimize and stabilize the training process, hyperparameters, and model architecture.

All models were trained on an NVIDIA DGX compute server on the Cambridge Consultants site with 8 NVIDIA V100 AI accelerator graphics processing units (GPUs). Each V100 can process approximately 10,000 replays, or 14 GB of replay data, per minute during training.

3.5 Confidence-aware defogger models

A variation of the 'standard' defogger model was also developed to predict confidence levels for each spatial unit count prediction in the model output. This was achieved by training the model to output a variance estimate indicating the variation in the training dataset for each opponent spatial unit prediction – the main count prediction in this case acting as the mean of a normal distribution with the predicted variance. This acts as a measure of the model's confidence in its estimated unit counts.

This additional output was implemented by increasing the output map channel count from 81 to 121 when active. The variance is not predicted for the player's spatial unit predictions, any non-spatial unit counts, or mining locations – hence the channel count increasing by 40, one for each of the opponent unit type count variances.

In order to train the outputs to represent the mean and variance of the unit count probability distribution, a modified loss function was used for the spatial unit-count-prediction loss, in place of the MSE loss. This approach is based on the Beta-Negative-Log-Likelihood loss introduced by [21], which allows the simultaneous learning of the mean and variance of the target distribution at the cost of introducing a new hyperparameter β_{conf}. This modified unit-count-prediction loss is given by:

$$Loss_{confidence} = \sum_{grid\ cells} \left\{ \hat{\sigma}^{2\beta_{conf}} \right\}_{no\ grad} \left[\frac{1}{2} \log log \left(\hat{\sigma}^2 \right) + \frac{(y-$$

Where $\hat{\mu}$ and $\hat{\sigma}$ are the predicted mean and variance, respectively, of the normal distribution for the unit count for a given unit type in a given grid cell.

As the player's spatial unit counts are part of the input, the variance of these counts could not be left as a prediction of the model, as its value tended to zero, causing the loss component for the prediction of the player's units to inflate exponentially and dominate learning. However, an using MSE loss for the player's units was found to adversely affect training, due to the mismatch between loss scaling between the player's and opponent's unit count predictions. A small, fixed variance was therefore used for the player's unit-count-prediction loss, allowing it to balance with the opponent unit-count-prediction loss once the predicted variance for opponent units had reduced. This fixed variance value therefore constituted an additional training hyperparameter. Despite this measure, the training remained less stable than the model which did not predict confidence, due to the sensitivity of the loss to changes in predicted variance and the balance between loss components.

As there is no ground truth for variance (as there is no variation in the unit counts for a given game state), it is not possible to directly assess the performance of variance prediction.

Instead, we plotted the mean predicted variance within bins of unit count prediction error magnitude across all frames of the test set replays (see Figure 3). Bins were defined such that there was a fixed number of bins evenly spaced logarithmically across the count error range, giving bin widths of ~ln(error) = 0.29.

On average, and above a count error of ~0.001 (i.e. for predictions in hidden areas) predicted variance was found to increase with increasing unit count prediction error, showing that confidence values do indicate the likely accuracy of unit count predictions.

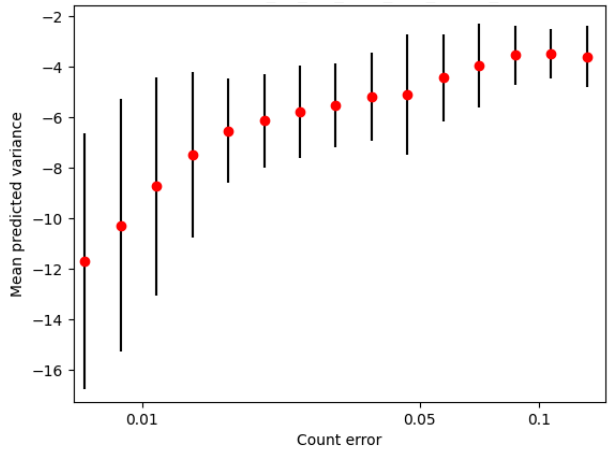

Figure 3. Unit count prediction error v. mean predicted unit count variance of the confidence model when applied to the test data set, showing the correlation between the two. Error bars show the standard deviation of the predicted variance values in each bin.

3.6 Performance results

We show quantitative performance metrics for the defogger model in Table 1. This table provides the values of 9 metrics calculated for our standard and confidence-aware defogger models, as well as 6 baseline approaches. Full details of each performance metric as well as all baselines are provided in Annex A.

Typically, the final standard (non-confidence-aware) model achieves the following performance:

- Over 30% IoU for placement of non-zero opponent unit counts in map regions obscured by the fog of war
- Over 70% IoU for spatial distribution of mining prediction in hidden regions

These results should be compared to the set of non-AI baselines (see Annex A) which achieve, at best for each metric:

- 26% IoU for placement of non-zero opponent unit counts in map regions obscured by fog of war
- 19% IoU for spatial distribution of mining prediction in hidden regions
- RMSE of 785 and 19 unit counts per unit type for non-spatial unit counts and integrated spatial counts

The model consistently outperforms all of the baselines at all periods of gameplay. For full results across all models and baselines for all performance metrics, see Table 1.

The prediction results of the confidence-enabled model are worse than the standard model across all metrics, likely due to the need for the model to estimate the additional confidence outputs, requiring it to distribute less of its representational capacity to the unit count predictions themselves. However, the confidence-enabled model still performs similar to or better than the baselines in many of the metrics. Notably, the IoU metrics are good whilst the spatial count error metrics are not as good, implying the spatial predictions of the main prediction head performed well but that that the predicted counts at each location were not as accurate. The performance of the non-spatial unit count prediction head was similar to the standard model, but

slightly worse, showing the predictive performance of the model as a whole was worsened by having to additionally predict opponent unit count variances.

Performance of both models was also assessed at different stages of the game, defined by blocks of 100 game frames; given that mode of play typically changes with absolute elapsed time, this provides a better picture of performance by game stage than using fractions of the total game length.

For the standard model, the locations of the opponent's units which are hidden from view are predicted very well initially (>60%), declining as the game progresses to <30% - however, the accuracy consistently outperforms that of the baselines at all game periods. The model predicts the locations of the player's units (which are visible in the input) nearly perfectly, and therefore matches the performance of baselines which replicate the input. The locations of the opponent's units which are visible are also predicted as well as possible based on the input alone, again equalling the baselines' performance. As with the overall results, the confidence-aware model performed worse than the standard model across all metrics at all game stages, but outperformed some or all baselines for some metrics, particularly IoU metrics. See additional results in Annex A. for a comparison of visible and hidden opponent unit prediction accuracy over the course of the game for the standard model and the baselines.

We highlight the main limitations of our approach:

- As the model has been trained on gameplay recorded from competitive players who display a high level of familiarity and skill with SC2, it can be difficult for the model to generalise to players who are less familiar with the game.
- Estimating the locations of enemy mobile forces has proven very challenging. Although the model is effective at locating static targets like bases, mining operations and static defence buildings, it has limited ability to estimate the location of unit concentrations.
- Our model does not make effective use of temporal sequence data. Although the architecture is designed to encode information from a sequence of input frames covering an extended time interval, we were not able to train the model to learn useful features from previous frames.

Model	IOU player	IOU opponent visible	IOU opponent hidden	IOU mining visible	IOU mining hidden	Unit count error player	Unit count error opponent	Unit count error non-spatial player	Unit count error opponent non-spatial
Standard model	**1.00**	**0.74**	**0.42**	**0.81**	**0.72**	**1.6**	**14.0**	**1.2**	**0.6**
Confidence model	0.97	0.68	0.31	0.70	0.19	187.2	191.6	1.7	1.6
Average baseline	**1.00**	0.71	0.02	0.94	0.00	**0.0**	24.1	791.1	245.1
LKP baseline	**1.00**	0.77	0.02	**0.99**	0.00	**0.0**	23.7	798.9	**243.5**
Mirror baselines	**1.00**	0.64	**0.26**	0.93	**0.19**	**0.0**	**19.1**	1442.8	102.4
Return baseline	**1.00**	**0.80**	0.01	**0.99**	0.00	**0.0**	24.1	792.3	244.7
Sum baseline	**1.00**	0.58	0.02	**0.99**	0.00	**0.0**	24.1	**785.3**	244.7
Zero baseline	0.00	0.00	0.00	0.00	0.00	25.6	25.6	1004.8	422.3

Table 1. Performance metrics for the 'standard' and 'confidence-prediction' defogger models, and the baseline models. The metrics for the best-performing model and baseline for each metric are highlighted in bold.

4 AI FOR DETECTING OPPONENT'S STRATEGIES

SC2 players may adopt different playing styles or "strategies" during a match, to improve their chances of victory. For example, they may attempt to produce large numbers of military units at the start of the game, in order to force a quick victory over their opponent. Alternatively, they may take a more defensive strategy, and focus on improving their economy so they can produce more late-game units (which are typically more powerful than early-game units). Players may also focus on building units of a specific type, e.g. airborne units, light infantry or armoured vehicles. Several of these strategies are commonly understood and named within the SC2 player community.

If a player knows what strategy an opponent is pursuing, they can take actions to counter this. For example, if their opponent is building large numbers of airborne units, they may focus on creating anti-air units.

This process was carried out in four stages:

- We extracted a set of input features from different game replays which were deemed to be likely to have different distributions depending on the opponent's strategy, based on knowledge of typical SC2 strategies.
- We used unsupervised clustering techniques to find collections of games where the opponent appeared to be following a common strategy, and manually labelled each collection.
- We trained a set of supervised classification models which could be used to predict an opponent's strategy at different stages of the game, using the clusters as "ground-truth" labels.
- We created a simplified strategy classification model, which used a smaller set of input features, to aid interpretability.

4.1 DATASET

For the strategic clustering work, we restricted the dataset to only the 100,450 replays available from SC2 version 4.10.0. We applied the same criteria as used for the defogger development to down select from these to 8,768 suitable replays. For the unlabelled strategic clustering process, all of these down-selected replays were used. For the supervised classifier training, the down-selected replays were divided into train, validate, and test sets according to an 80:10:10 ratio.

4.2 INPUT FEATURES

For these studies, we selected a range of game features which fulfilled two key criteria. Firstly, they were deemed likely to be useful indicators of an opponent's strategy, and likely to have different distributions for different strategy types. Secondly, they had simple definitions which could be easily interpreted by a human. . Furthermore, for the supervised strategy classifier we only used features which could be calculated while a game is in progress, allowing strategies to be detected during the game rather than after its conclusion. The features we chose included:

- The total numbers of different types of units (infantry, armoured vehicles, or airborne vehicles) and buildings, and the amount of resources spent on each.
- The total amount of resources which have been collected and spent, and the recent rates of collection and spending.
- Whether certain unit and building upgrades have been built during the game up to the current time.
- The time at which certain buildings were first constructed (if constructed yet).
- The distance between buildings and the player's starting location.
- How much time the game has been going on for.

To calculate these features, we used ground-truth information from replays, extracted using the methodology outlined in Section 3.2. After producing our final strategy classifier model, we also extracted these features from the predicted full-state outputs of the defogger model (see Section 3) This allowed us to assess the impact of the imperfect reconstruction of the full state on the strategy classifier model's performance (see Section 4.7).

4.3 UNSUPERVISED CLUSTERING OF STRATEGIES

After extracting features for strategy classification, we sought to identify clusters of related games in the data. To do this, we used the popular K-means clustering algorithm to identify 7 strategic clusters in a feature space defined by 8 strategically relevant features, selected using expert knowledge of the SC2 domain to correlate with the player's strategic intent while

minimizing dependence on the details of how the game played out.

After the replays were each assigned to one of these strategic clusters, we examined the distributions of the input features and watched samples of replays from each cluster, to determine the kind of strategy to which each cluster corresponded. These assignments were based on the authors' existing knowledge and online research of strategies which SSC2 players typically pursue. The strategy labels applied to the 7 clusters, along with the number of replays in each cluster, are:

- **Bio aggressive (2113 replays)** – a high focus on the production of "bio" (or infantry) units, with limited spending on vehicles. Players typically attack early and delay economic expansion.
- **Air (1603 replays)** – a high focus on the production of the Battlecruiser and Viking air combat units, with correspondingly high spending of the Vespene gas resource required to construct these units.
- **Bio macro (2195 replays)** – a high focus on the production of bio units, but with a higher economic focus than the 'bio aggressive' strategy, and more production of supporting units like Medivac air vehicles.
- **All-in (547 replays)** – an extremely low focus on economic expansion, and very early attacks using armies of bio units in an attempt to force an early win before an opponent has built defensive capabilities.
- **Harass/proxy (330 replays)** – a high focus on the production of "raiding" bio units such as the Reaper. The player often builds a "proxy barracks" close to the opponent's base to enable a quick attack through construction of units nearby.
- **Mech (1739 replays)** – a high focus on the production of mech units (armoured vehicles), with a correspondingly high number of buildings to produce these vehicles. This leads to games of medium length.
- **Turtle (176 replays)** – a high focus on defensive buildings, to guard the player's base. This can lead to very long games. This is a relatively unusual strategy.

4.4 Supervised classification of strategies

After assigning each game in the dataset to one of these strategy clusters, we trained a supervised classifier so we could predict an opponent's strategy in an unseen game from outside the clustering dataset, and while the game is in progress. During training experiments, we assumed each game's strategy label at the end of the game was a "ground truth" label of the opponent's strategy. We calculated a set of 68 input features, calculated at 100 randomly sampled frames from each game in the data. A sub-set of 10% of the games were set aside as a validation set, used to assess the classifier's performance when selecting an algorithm and tuning its hyperparameters. A further 10% of the games were set aside as a test set, to calculate the unbiased performance of the final model.

After experimenting with a support vector machine (SVM) technique, we used a feed-forward neural network classifier due to its improved performance and training time relative to an SVM. The network included two hidden layers of 256 nodes each, with an Exponential Linear Unit activation function following each node [22] . The outputs of the final layer were passed through a softmax function, such that the model output would be the probability of a data entry (i.e., the replay from which the input features were taken) belonging to each strategy class. The network was trained to minimize the cross-entropy between the ground-truth strategy and each of these output probabilities, using the Adam optimization technique [23] with a batch size of 128 samples and a learning rate of 10^{-4}. Dropout was used to mitigate overtraining, and data from smaller strategy classes was oversampled to improve the classifier's performance on these classes. To improve the numerical stability of the neural network, each feature was transformed so that training data was distributed with a mean of zero and a standard deviation of 1. Linear transforms were used for most input features, while a small number were log-transformed.

Across the whole test set, this classifier showed an accuracy of 0.57, with the precision and recall on each strategy class ranging between 0.34 and 0.66 (see Table 2). The classifier performed best on classes with large numbers of members such as Bio/macro, and poorest on classes with fewer members such as Turtle.

Strategy	Recall	Precision
Bio/aggressive	0.64	0.47
Air	0.46	0.87
Bio/macro	0.66	0.61
All-in	0.34	0.49
Harass/proxy	0.48	0.52
Mech	0.58	0.56
Turtle	0.35	0.41

Table 2. Performance of the strategy classifier neural network for each of the 7 strategic clusters.

We examined the classifier's performance as a function of game time, as shown in Figure 4. The classifier's performance at the early stages of games is relatively poor with an accuracy score of 0.30 during the first two minutes of play. This is because unit compositions and buildings constructed by players pursuing different strategies do not differ significantly at early stages of the game. The performance then improves as the game progresses, and unit compositions diverge for different strategies. For example, for gameplay after the ten-minute mark, the accuracy score reaches 0.81. The classifier performed best on long games.

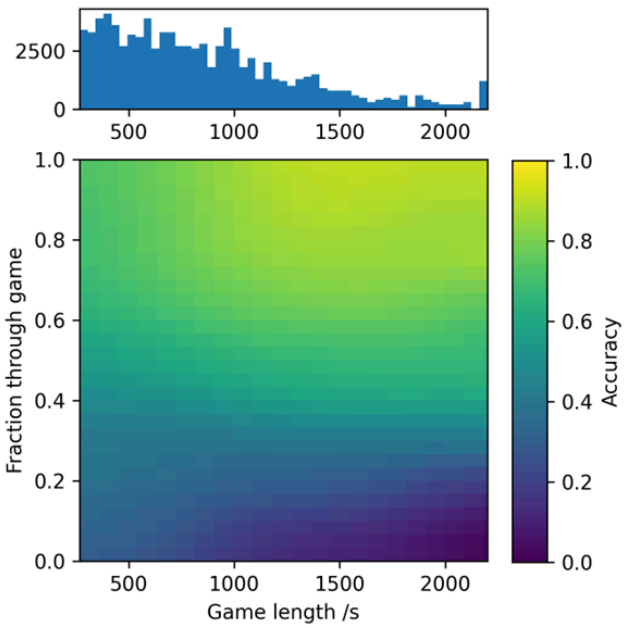

Figure 4. The performance of the strategy classifier for games of different length (x-axis), and at different stages through the game (y-axis). The histogram at the top shows the number of games with different lengths.

4.5 Creating a simplified classifier

While the neural network strategy classifier had satisfactory performance when predicting an opponent's strategy, it proved difficult to interpret the reasons for its predictions. This was due to the large number of input features, many of which had relatively non-intuitive definitions.

To resolve these issues, we created a simplified classifier neural network model, which used only the 15 features which we judged to be most easily interpretable by a user. These features mainly consisted of the number of key buildings and units belonging to the opponent, and the proportions of resources spent on key unit types.

We wanted the predictions of this model to mimic the original classifier as closely as possible. Therefore, we used a model distillation technique, where the simplified "student" model is trained to match the outputs of the original "teacher" model, rather than the ground-truth labels in the data. Apart from the reduced number of inputs to the 15 simplified input features, the student model's architecture was identical to that of the teacher model. It was trained to minimize the mean squared error between the logits for its strategy predictions and the logits for the teacher model's predictions. We considered a more complex loss function which included a term rewarding correct predictions of ground-truth strategy labels [24]. However, we ultimately used the simpler mean-squared error method, because our priority was for the student model to match the teacher's predictions, rather than optimally matching ground-truth labels. The same training dataset and optimizer were used as when training the teacher model.

After training concluded, we measured the correlation between the student and teacher models' predictions using a Pearson correlation coefficient, finding values of 0.96 or greater for each strategy class, on the test dataset. The distilled model showed an accuracy of 0.53 relative to ground-truth labels across the full test set, close to the original model's accuracy of 0.57. As for the original model, the performance varies with game time, with an accuracy score of 0.22 in the first two minutes, and 0.81 for gameplay after ten minutes.

4.6 Explainability with Shapley Additive Explanations

Following the training of the simplified strategy-classification model, we considered how to create explainability outputs for it. Using this simplified model, we created an explanation model using the Shapley Additive Explanations (SHAP) method [25]. This model apportions values to each input feature for a given classifier prediction, showing the amount by which that input feature is likely to have increased or decreased the probability of that prediction. For example, if an opponent has a large number of airborne units, this would typically indicate that they are pursuing the "Air" strategy, meaning the model will give a large SHAP value for the air spending fraction and similar features. This provides some insight into the key information which the strategy classifier is using to calculate its predictions, helping the user to judge whether these predictions are reliable or not. We used the DeepSHAP method, which uses optimizations to improve the speed of SHAP

calculations with neural network-based classifiers.

4.7 Performance using defogger outputs

The strategy clusters and classification models were trained using ground-truth information about the opponent's units and resources. In a more realistic setting, such information would not be accessible to a player, meaning they would need to only the partial game state information available during normal play. It follows that they could therefore use the outputs of the defogger model to estimate the full game state from the available partial game state, to use as input into the strategy classifier. We therefore measured the performance of the simplified classifier when its input features were calculated from the spatial maps and unit count vectors output by the defogger model. The majority of the features (such as number of units and times they were first constructed) could be directly extracted from these outputs, or derived using simple methods (e.g. using the difference between unit count estimates in adjacent game frames to estimate when units are built or destroyed). However, resource-related values are not directly predicted by the defogger model. Resource collection values therefore needed to be estimated using the number of resource-gathering units predicted by the defogger model, together with the typical rates of resource gathering by these units as seen in games in the defogger's training dataset. Rates of resource spending could be estimated from the number of units being built (as predicted by the defogger), and their costs.

After calculating these input features, we measured the performance of the strategy classifier model as the values of each input feature were changed from their ground-truth values to values derived from the defogger model (see Figure 5). We used the F1 score to measure this performance (the harmonic mean of precision and recall), macro-averaged across the different strategy classes (so that the metric assigns equal importance for each class). The order in which these features were converted was chosen to minimize the rate at which the performance dropped (i.e. picking the feature which gave the lowest performance drop, at each step moving from left to right)

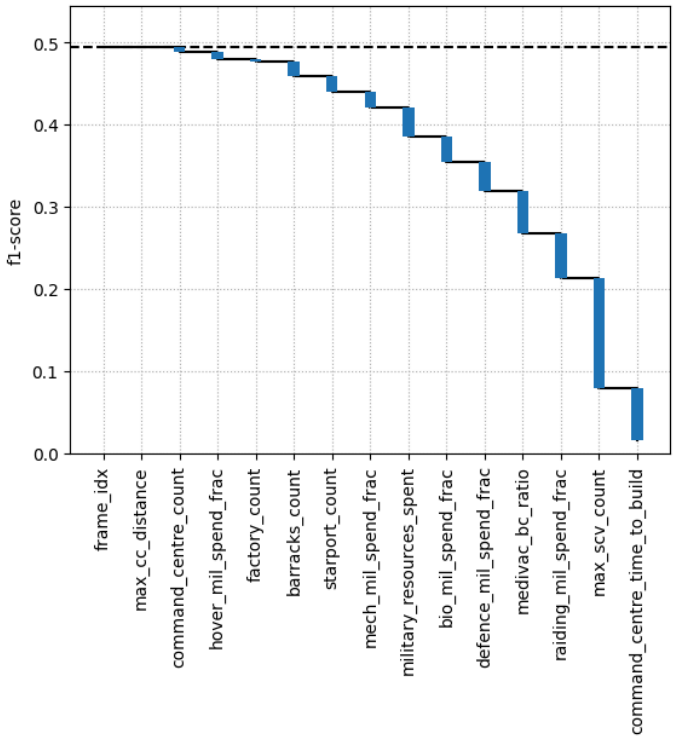

Figure 5. degradation of strategy classifier performance as different features are changed from ground-truth to defogger-derived values (moving left to right).

The classifier had an initial F1 score of 0.49, which dropped to just 0.015 after all features were replaced. At this point, the classifier predicted opponents in all games were following the "All-in" strategy. However, some of the features could be converted to defogger-derived values without substantial drops in performance, such as the maximum distance of buildings from the command centre, or the number of command centres the opponent has. This suggests that in a more realistic setting, the user could focus on gathering accurate information about the variables which gave substantial performance drops, such as the number of resource-gathering units or the time the opponent took to build a second command centre, while using the defogger-derived values for features which give a smaller performance drop.

We also applied some post-processing to the defogger model's outputs, to make them more closely match real data, to see if this improved classifier performance. We re-scaled the unit count vectors to match the distributions in ground-truth training data, and rounded unit counts down to the previous integer. This slightly improved the performance of the strategy classifier after all features were converted to defogger-derived values, increasing the F1 score from 0.015 to 0.065. However, this performance is still too poor to be useful to a player. As a future development, the strategy classification

model could be re-trained using defogger-derived training data, so it can learn the patterns of noise which the defogger introduces. This is likely to improve the strategy classifier's performance when applied to defogger-derived data.

5 The "bridge" to provide the right information and the right explanation

As mentioned in section 2, there is disconnect between AI outputs and providing explanations for non-AI-experts about the process behind those AI outputs. Numerical locally interpretable model explanations such SHAP [25] are helpful, however they require specialised AI expert knowledge to fully understand.

Recent improvements in natural language generation have made rationalisation an attractive technique for AI explanations, because it is intuitive, human-comprehensible, and accessible to non-technical users. Rationalisation provides explanations justifying a model's prediction in natural language [26][27]. It aims to offer a coherent and interpretable reason for that prediction and allows individuals without domain knowledge to understand how a model arrived at a prediction.

For these reasons, we used rationalisation as our approach to breaching the technical gap between outputs and users' understanding. In our explanations, we constructed rationales or justifications which present the input features influencing the model's prediction. Our objective was that the reasoning behind the prediction could be understood by a non-technical user simply by reading the explanation/rationale, thereby revealing the model's decision-making process.

We followed the model of 'selective explanation's [28]: usually, users do not expect an explanation can cover the complete cause of a decision - instead, they desire an explanation that can convey the most valuable information that contributes to the decision. A sparse explanation, which includes a minimal set of important features that help justify the prediction, is preferred.

Our aim was to answer the important questions which are key to the user achieving their goal:

- What might the opponent be doing (behind the fog of war)? What can an AI tell me about red-force positions through information with limited availability, and with what level of confidence?
- What do the opponent's actions mean at different stages of the game? What can an AI tell me about the red-force strategy, so that I can understand and incorporate this information into my decision making accordingly?

We answered the questions above by creating a UI that displayed AI outputs with graphical and textual explanations. The explanations were expressed in gaming language and translated to a military language to test transferability of the technique to C2 environments. Our usability study (see Section 6) then highlighted the areas that the users (players) thought would best inform their decision making. Descriptions of these are found in the following sections.

5.1 User interface

The User interface is segmented into three primary views: Player view, AI Vision, and Opponent's Predicted Strategy. Each segment provides distinct functionalities essential to enhancing strategic planning and decision making.

Player View: This segment serves as the live link between the ongoing gameplay and the user interface, displaying real-time information to the player, see Figure 6. It has the flexibility to be hidden, allowing the player to focus on the AI output.

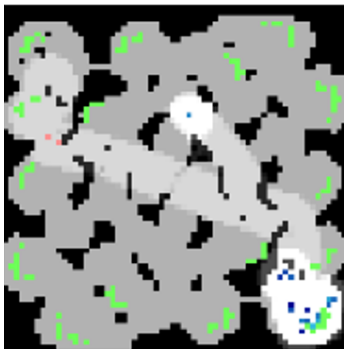

Figure 6. Player's minimap, showing what is normally the player's view of the game and fog of war(areas in dark grey are unexplored, areas light grey reflect those where "scouts" have been but which are no longer within sight range of the player's units)

AI Vision: The AI Vision section is a crucial overview of the game environment, predicting enemy location and unit counts, in order to aid players in resource allocation and tactical planning. It incorporates several key elements (see Figure 7):

- **Enemy Location Prediction:** estimated positions of opponent units output by the defogger model.
- **Confidence Level:** indicates the certainty of the defogger model's predictions of opponent units.
- **Ground Truth:** Displays true positions of

enemy units, to validate AI predictions.

- **Predicted Unit Count:** Offers an estimate of the total number of opponent units across the whole map, as output by the defogger model.

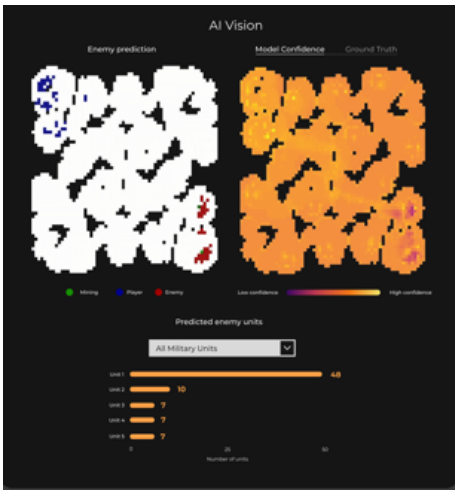

Figure 7. Screenshot of the UI showing opponent's units through the fog of war as predicted by the defogger model, prediction confidence levels, and total predicted unit counts across the map.

Opponent's Predicted Strategy: This segment focuses on anticipating the opponent's future moves and strategies, see Figure 8.

- **Prediction and Tactics:** Prediction of the most likely opponent strategy from the simplified strategy classifier model, and suggestions of their likely tactics based on this strategy.
- **Prediction Timeline:** Line graph showing the predicted probability of each opponent strategy over time.

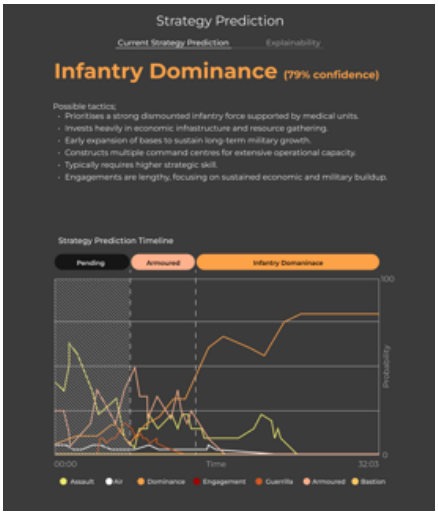

Figure 8 Screenshot of the UI showing strategy prediction, description of strategy and strategy prediction timeline.

- **Explainability:** Provides insights into the rationale behind opponent strategy predictions, helping players understand and trust the AI's strategic assessments, based on feature values, see Figure 9. We did not include SHAP-related language to reduce the knowledge barrier for new users.

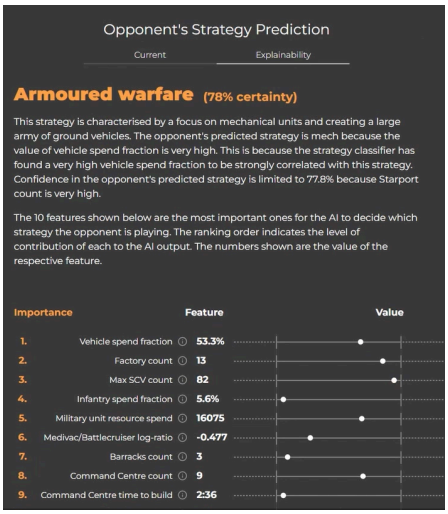

Figure 9 Screenshot of the UI showing user-centric explainability of the opponent strategy prediction

The interface employs military language where possible, to ensure clarity and precision, aligning with the terminology familiar to the C2 user base. Additionally, an information-hover feature allows users to obtain detailed descriptions of various elements by hovering over the mouse over them, facilitating a deeper understanding without cluttering the interface. The combination of the segmentation of the UI and detailed functionality enhance the strategic depth and user engagement with the application.

6 User feedback on AI-decision support systems

Critically evaluating AI-based systems and user interfaces, and their usability and level of trust in the context of decision support is important in informing future interface design. We therefore carried out a user feedback study to assess the system developed as part of this project.

The study protocol is laid out in Annex B, Figure 12. The main research questions we wanted to address using user feedback were:

1. What effect does explanation have on users' satisfaction and trust in the AI-decision support system?
2. What type of explanation – textual or graphical - is most useful for a positive effect on satisfaction and trust?

The usability constructs were measured using the system usability scale (SUS) questionnaire [29], and measures of trust were adapted from a 'cruise control' system

evaluation questionnaire **Error! Reference source not found.**. To supplement the questionnaires, we conducted a semi-structured interview to explore the transferability of our tested models and interfaces to operational planning, and to do a deeper dive on trust. The questions can be found in Annex B.

Five video segments were chosen to highlight different aspects of the AI model predictions and subsequently the different user interfaces:

- Segment 1: from 'pending' (no strategy prediction) to the first strategy prediction.
- Segment 2: determining confidence for first prediction of an 'Armoured Warfare' strategy of approx. 40%.
- Segment 3: increasing confidence in the first prediction of an 'Armoured Warfare' strategy from 70% to 90+%.
- Segment 4: switching between 'Armoured Warfare' and 'Air Dominance' strategy predictions, where both wavered around 50%.
- Segment 5: establishing confidence for 'Air' strategy prediction from 60% to 90+%.

6.1 Background questionnaire results

The background questionnaire was provided before the users began providing feedback on the system. In total, 10 users provided feedback, with varied experience playing SC2. All users considered themselves to be either experts (7 people) or somewhat experts (3 people) at computer games and all users had experience playing real-time strategy games.

6.2 Usability and trust questionnaire results

System Usability Scale

The System usability scale (SUS) [30] is one of the most used and well-known scales for assessing usefulness of a UI in UX research. With high reliability and validity, the SUS consists of a series of 10 Likert-scale statements that produce a score between 0 and 100 (100: most useful and 0: not useful at all). Using this score, we can get a single numerical value that summarises the perceived usefulness of a given UI.

Overall SUS scores: Using a threshold of 60 to mark "useful" amongst the SUS scores (Figure 10), the graphical and control UIs were seen as useful in time segments 2 and 4, where the confidence is determined for the 'Armoured Warfare' strategy prediction and the prediction switched from this to the 'Air Dominance' strategy. Interestingly, although the text UI appeared *least* useful when categorized by positive and negative statements, it appeared *more* useful than the control and graphical UI in both time segment 3 and 5, where there is an increase in the confidence in the 'Armoured Warfare' prediction from 70 to over 90% and establishing confidence for the 'Air Dominance' prediction.

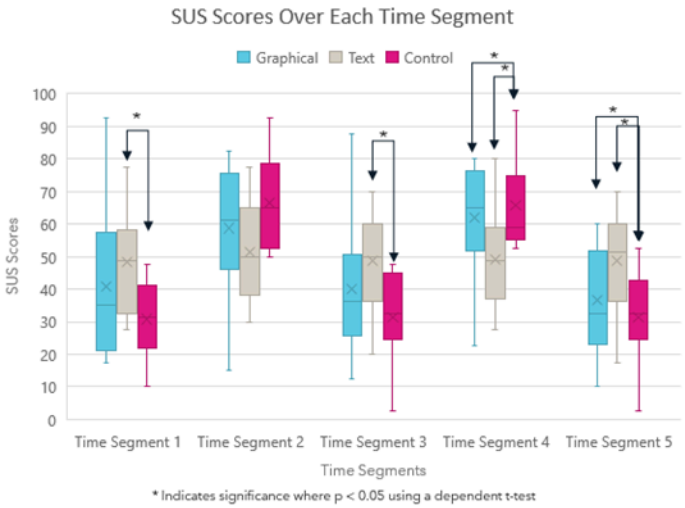

*Figure 10 . System usability scores over each time segment for each stimulus. Note: * indicates a statistically significant (p<0.05) result between two stimuli using a dependent t-test.*

Trust Assessment

To assess trust, we used a validated subjective metric that was short and generalisable. The authors in [31] noted that trust can be narrowed in scope from a general overarching theme of trust to reliability, predictability, and efficiency. We modified their original trust scale (used to assess trust of vehicles with cruise control systems) to match that of a gameplay task. We interpreted scoring as 'the higher the summed value of the four Likert questions, the more the user trusted the system'.

Predictability: All three UIs were similarly and consistently rated for predictability (except for time segment 1 where the control UI was slightly more predictable).

Reliability: The graphical UI consistently had ratings above 3 for each segment. The control UI had started off with similar reliability ratings to the graphical UI, but scores decreased steadily as each time segment went on, e.g. to an average rating of 2.1 for time segment 4.

Efficiency: The graph UI received the highest average scores for all time segments. However, in general, none of the UIs were considered efficient for playing SC2, understandably due to lack of integration with the game.

Trust (overall): The control UI was rated as being the least trustworthy UI, even though it was the simplest. Both the graph and the text UI were rated as more trusted, with the graph UI having ratings above 3 throughout.

6.3 Semi-structured interview results

A semi-structured interview is a qualitative research method with a pre-determined set of open questions and with opportunities for the interviewer to explore specific themes or responses further [32]. The interview topics covered feedback on the different types of user interfaces, whether accuracy influences trust in AI, and transferability of the AI models and user interfaces to operational planning. The questions are listed in Annex B.

We analysed the interview transcripts using thematic analysis supplemented by an LLM (a local version of GPT4o), and validated by a human. We based our thematic analysis methodology on two papers [33][34], picking the results which best grouped feedback while reflecting the nuances of different users. The following themes emerged:

Feedback on user interfaces. In accordance with the SUS analysis, the majority of users preferred the graphical UI for its quick readability and efficiency of information delivery, especially in fast-paced gaming scenarios. Some users also liked the minimalist control interface with no explainability components, highlighting the balance between providing useful information and overwhelming users with too much text. Users commented that the UI would be more useful as a tool to review their gameplay, rather than for real-time decision making.

Perceived trust influenced by overall model accuracy and usefulness. Trust is closely tied to how useful the information being provided is, and strongly informed by lived experience of using the model. A single statement or statistic provided to the user (e.g. “the model is 68% or 88% accurate overall”), did not significantly change trust perceptions. If the user believed that the model was usefully predicting future events, then they were more likely to trust the system and base their decisions on it.

Trust in AI for use in real-world applications. Users agree on the potential of AI but remained cautious, especially in high-stakes situations. They expressed reservations about fully trusting AI, preferring it as a supplementary information source rather than a primary decision-making tool. Users do not want to be told exactly what to do, but instead be provided with either a summary report or predictive information to enhance their situational awareness while retaining autonomy over decision-making.

Personalisation. Users mentioned that the user interfaces would benefit from personalisation along the dimensions of player skill level, expertise, and game progress. For example, actionable insights (based on the assumption of the highest confidence strategy) would benefit novice users; while expert users would prefer seeing all possible predicted opponent strategies, however unlikely, to mitigate risks. This is because higher-level users expected deception by their opponents, drew on more sources of information to make their decisions, and had the cognitive capacity to deal with low-probability events.

7 Conclusions and Recommendations

In this paper, we have used the commercial video game SC2 as a medium for simulating battle environments to advance AI research relevant to military C2 operations. We have taken a human-centric approach to AI development, understanding what a user (or in this case a player) would want to get from an AI to assist and augment their decision making.

We developed two AI approaches to meet user technical requirements: (i) piercing the fog of war to infer an opponent’s assets’ locations and (ii) opponent strategy detection and strategy changes over time.

We assumed that the human (player) did not have technical knowledge of AI and therefore considered their needs for understanding of AI and AI outputs for decision making.

Finally, we also used the gaming environment as a simulation medium for human-AI interaction. We explored through a usability study what humans (players) thought of the information provided, and their feedback on the usability of the display of graphical and textual explanations.

In conclusion, our project provides the following benefits for advancing military research:

- **A user-centric approach**. AI is not developed in isolation. Insights on user needs for information and functionality need to be understood in order to develop the right AI solution. A user-centric approach should be applied to real military projects in order to direct resources towards the most valuable developments of AI.
- **Piercing the fog of war**. Military operations and planning at operational and tactical level encounter the challenge of partial information for decision making. Our approach illustrates how AI could aid this challenge with a “defogger” approach that ingests a huge amount of historical data and trends in order

to nowcast what might be happening on the situation at hand.

- **Narrowing down key features for characterising opponents' movements and behaviours**. The distilled AI classifier could perform with only the top 15 most important features for strategy classification. This has huge value for military intelligence in situations in which it is difficult to focus on the vast amounts of varied information. By narrowing down to key features for a particular set of events that define, for example, a strategy, the data used and the speed of insight generation can be optimised and delivered faster to the decision maker.
- **Flexible user-centric explainability framework**. Explainability of AI is one of the most looked for AI developments. Non-AI experts and AI users need to understand and trust the outputs of an AI agent. In our project, we focused on one of the most evolved explainability techniques, SHAP, to detect feature importance for the output of the classifier. We explored what type of explanations would be most useful to the user and how to translate the outputs of SHAP into a language that could be understood. This is, and will be, a key element for any implementation of AI within the military environment, where it is crucial that military experts can understand and base their decisions on information with confidence.
- **Human-AI interactions.** Human-AI interactions are becoming extremely important, as AI will likely be deployed as part of a socio-technical environment (within a team, within a unit, etc.). Our approach to explain and display information in a way and language that is easy for the user to absorb, within a user interface that is tailored to the task, is also beneficial to consider when deploying solutions in a military context. This makes the output of the AI development useful and valuable to the user, allowing it to be effectively incorporate this into their decision making.
- **User feedback and feedback loops**. User feedback and feedback loops are also essential to refine and develop effective human-AI interactions. We developed a usability study that looked at :i) SUS scores through structured questionnaires and ii) in depth interviews for qualitative insights. This provided us with valuable insights that would feed back into our design.

In conclusion, recommendations of further work include:

1. Push the technology closer to real-world military applications: (i) move from gaming simulation environments to more realistic simulators (ii) address data volume and quality issues inherent in realistic applications.
2. Expand Human-AI interaction research and development within military contexts: explore human-AI interfaces, communication channels and interactions that include different ways of communicating AI insights to military users.
3. Build user trust for fast implementation and uptake of AI solutions for C2: (i) implement user-centric explainability frameworks (ii) build user trust for fast implementation of AI solutions for C2 decision making. Insights on usability, trust and personalisation where extremely valuable to assess how our explanations and UI could be refined.

8 Acknowledgements

The research reported on in this paper was funded by the UK MOD Machine Speed Command and Control (MSC2) project. This project was part of the UK Defence Science and Technology Laboratory's (Dstl) AI Programme with the intent to transform C2 by enabling more `timely and effective C2 processes across all environments, domains and levels of command, so the Defence enterprise can anticipate and adapt more successfully than adversaries.

This paper is one of six presented at the 29th ICCRTS which document different aspects of the MSC2 project which explored the feasibility of a Human Agent Collective (HAC) that combines human insight with machine speed AI agents employing shared digital artefacts, shifting C2 from human teams to human-machine teams, where humans and AI work together. We would like to thank Dr. Stephen Helsdon and Dr. Martyn Fletcher for their invaluable direction and feedback.

9 References

[1] Tsavdaridis, G., Koukoutsis, E. and Karadimas, N.V., 2019, December. Techniques Handling Operational Information During Military Decision Making Process. In 2019 3rd European Conference on Electrical Engineering and Computer Science (EECS) (pp. 117-122). IEEE.

[2] Bjola, C., 2022. Artificial Intelligence and Diplomatic Crisis Management: Addressing the "Fog of War" Problem. VALENTIN NAUMESCU RALUCA MOLDOVAN, p.25.

[3] Tryhorn, D.N., 2021. Exploring Fog of War Concepts in Wargame Scenarios.

[4] Randall, P., S., 2015, Strategy and the role of the enemy, Infinity Journal, Volume 4, Issue 3, pages 28-32

[5] Nashed, S. and Zilberstein, S., 2022. A survey of opponent modeling in adversarial domains. Journal of Artificial Intelligence Research, 73, pp.277-327.

[6] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. and Chatila, R., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion, 58, pp.82-115.

[7] Liao, Q.V. and Varshney, K.R., 2021. Human-centered explainable ai (xai): From algorithms to user experiences. arXiv preprint arXiv:2110.10790.

[8] Springer, A. and Whittaker, S., 2019. Making transparency clear. In Algorithmic Transparency for Emerging Technologies Workshop (Vol. 5).

[9] Liao, Q.V. and Varshney, K.R., 2021. Human-centered explainable ai (xai): From algorithms to user experiences. arXiv preprint arXiv:2110.10790.

[10] Futia, G. and Vetrò, A., 2020. On the integration of knowledge graphs into deep learning models for a more comprehensible AI—Three challenges for future research. Information, 11(2), p.122.

[11] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P. and Oh, J., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), pp.350-354.

[12] Gehring, J., Ju, D., Mella, V., Gant, D., Usunier, N. and Synnaeve, G., 2018. High-level strategy selection under partial observability in starcraft: Brood war. arXiv preprint arXiv:1811.08568.

[13] 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. Advances in Neural Information Processing Systems, 31.

[14] Jeong, Y., Choi, H., Kim, B. and Gwon, Y., 2020, April. 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, No. 04, pp. 4296-4303).

[15] https://develop.battle.net/documentation/starcraft-2/game-data-api

[16] Ronneberger, O., Fischer, P. and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing.

[17] https://pytorch.org/

[18] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[19] Smith, L.N. and Topin, N., 2019, May. Super-convergence: Very fast training of neural networks using large learning rates. In Artificial intelligence and machine learning for multi-domain operations applications (Vol. 11006, pp. 369-386). SPIE.

[20] Smith, L.N., 2018. A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820.

[21] Seitzer, M., Tavakoli, A., Antic, D. and Martius, G., 2022. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. arXiv preprint arXiv:2203.09168.

[22] Clevert, D.A., Unterthiner, T. and Hochreiter, S., 2015. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

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

[24] Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[25] Lundberg, S.M. and Lee, S.I., 2017. A unified

approach to interpreting model predictions. Advances in neural information processing systems, 30.

[26] Atanasova, P., 2024. A diagnostic study of explainability techniques for text classification. In Accountable and Explainable Methods for Complex Reasoning over Text (pp. 155-187). Cham: Springer Nature Switzerland.

[27] Gurrapu, S., Kulkarni, A., Huang, L., Lourentzou, I. and Batarseh, F.A., 2023. Rationalization for explainable nlp: A survey. Frontiers in Artificial Intelligence, 6.

[28] Miller, T., 2019. Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, 267, pp.1-38.

[29] Coppers, S., Van den Bergh, J., Luyten, K., Coninx, K., Van der Lek-Ciudin, I., Vanallemeersch, T. and Vandeghinste, V., 2018, April. Intellingo: An intelligible translation environment. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-13).

[30] Brooke, J.B. (1996). SUS: A 'Quick and Dirty' Usability Scale.

[31] Cahour, B. and Forzy, J.F., 2009. Does projection into use improve trust and exploration? An example with a cruise control system. Safety science, 47(9), pp.1260-1270.

[32] Jamshed, S. (2014). Qualitative research method-interviewing and observation. Journal of basic and clinical pharmacy, 5(4),87.

[33] De Paoli, S., 2023. Performing an Inductive Thematic Analysis of Semi-Structured Interviews With a Large Language Model: An Exploration and Provocation on the Limits of the Approach. Social Science Computer Review, p.08944393231220483.

[34] Dai, S.C., Xiong, A. and Ku, L.W., 2023. LLM-in-the-loop: Leveraging large language model for thematic analysis. arXiv preprint arXiv:2310.15100.

Annex A – Defogger training details

10 Defogger hyperparameters

The full sets of hyperparameters used for training the final defogger models are shown in Table 1.

Hyperparameter name	Description	Standard model value	Confidence model value
Batch size	Number of game sequences per minibatch	128	128
Sequence length	Number of frames per input sequence	1 (see limitations)	1 (see limitations)
Number of batches trained	Total number of training steps	99,000 (approx. 200 epochs)	150,000 (approx. 300 epochs)
Initial learning rate	Learning rate at start of training	0.0005	0.001
Peak learning rate	Max learning rate during training	0.005	0.005
Peak learning rate position	Fraction of the way through training that the peak learning rate occurs	0.3	0.2
Mining loss scale	Scale factor for mining prediction loss term (α_0)	0.01	0.01
Auxiliary global unit loss scale	Scale factor for the auxiliary global unit count prediction loss term (α_1)	$2x10^{-5}$	$2x10^{-5}$
Opponent unit loss scale	The unit prediction loss for opponent units is scaled up by this factor compared to the loss for friendly units	1.0	10.0
Gradient clip value	All gradients are clipped to +/- this value for stability purposes	0.1	0.1
Minimum variance	Minimum allowed value for total output variance	N/A	0.0001
Friendly unit count variance	The variance used for friendly unit counts in the loss formula (not learned)	N/A	0.0001

Table 1. Final training hyperparameters

11 Defogger performance metrics

Performance metrics

A number of metrics were used to assess different aspects of the model's performance, given its multi-faceted output. These can be divided into a set of categories as follows:

- **Intersection over Union (IoU)** – spatial performance metric assessing how well predicted unit locations match the ground truth:
  - **Player unit IoU** – Average IoU over all player unit types
  - **Opponent unit IoU** - Average IoU over all opponent unit types
  - **Opponent visible unit IoU** – IoU only of the opponent's visible units
  - **Opponent hidden unit IoU** – IoU only of the opponent's hidden units
  - **Mining IoU –** IoU of mining locations (i.e. locations of Command Centres, Orbital Commands and Planetary Fortresses)
  - **Mining visible IoU** – IoU only of mining locations in visible regions of the map
  - **Mining hidden IoU** – IoU only of mining locations in hidden regions of the map
- **Unit count error –** root mean squared error (RMSE) in the predicted count across all unit types:
  - **Player unit count error** – RMSE of the unit count for the player's units
    - ●.o.●.1 **Spatial** – Player unit count error for the spatial unit prediction head
    - ●.o.●.2 **Non-spatial** - Player unit count error for the non-spatial unit prediction head
  - **Opponent unit count error**
    - ●.o.●.1 **Spatial** – Opponent unit count error for the spatial unit prediction head
    - ●.o.●.2 **Non-spatial** - Opponent unit count error for the non-spatial unit prediction head

12 Defogger baselines

The baseline models used alongside the defogger model were as follows:

- **Zero** – lower-limit baseline which returns zeros in all outputs, i.e. no predicted units or mining locations, ignoring all input. This baseline is useful to show that the model is making any correct predictions, including of known information.
- **Return** - returns the most recent game frame as the output, i.e. a direct copy of the input unit type counts and locations The locations of any opponent Command Centres (including snapshots) in this unit count are returned as mining locations.This baseline is useful for checking that the model is predicting known information correctly.
- **Sum** - sums the input data over all input game frames and normalises over the input frame count. The mean unit count for each unit type over all input frames is therefore returned. The locations of any opponent Command Centres (including snapshots) in this mean unit count are returned as mining locations. This baseline therefore represents a very simple use of all frames in the input sequence.
- **Average** - finds the average number of units of each type across the input frames and returns the last position of the units present for the longest up to that average count. The locations of any opponent Command Centres (including snapshots) in this average unit count are returned as mining locations. This baseline represents a slightly more intelligent method of making use of the full input sequence.
- **Last-known Positions (LKP)** - attempts to track unit units' movements by associating unit positions frame-to-frame. Returns the last known position of each unit, unless it appears to have been destroyed – units which entered the fog of war are kept. The locations of any opponent Command Centres (including snapshots) in this unit count are returned as mining locations. This baseline represents a variation on making use of the full input sequence.
- **Reflect the Player** - reflects the unit types and positions of the player across the map according to the symmetry of the map, taking into account observed enemy units. Visible opponent units are returned as observed and subtracted from the total count of the same type to be mirrored. Units which would be mirrored into visibly empty areas are returned at the nearest non-visible location. The locations of any opponent Command Centres (including snapshots) in this mirrored unit count are returned as mining locations. This baseline is based on the assumption that players carry out similar actions at similar times during the game.

13 Additional defogger results

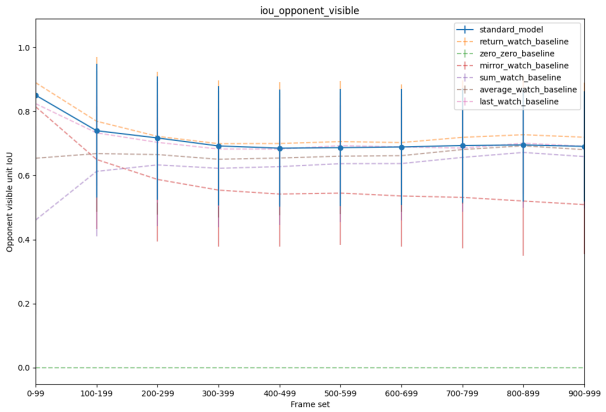

Figure 11 IOU opponent visible

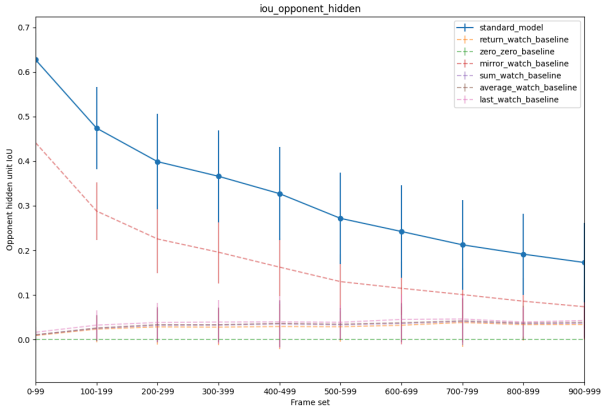

Figure 12 IOU opponent hidden

Annex B- User feedback study on AI-decision support systems

Graphics explaining:

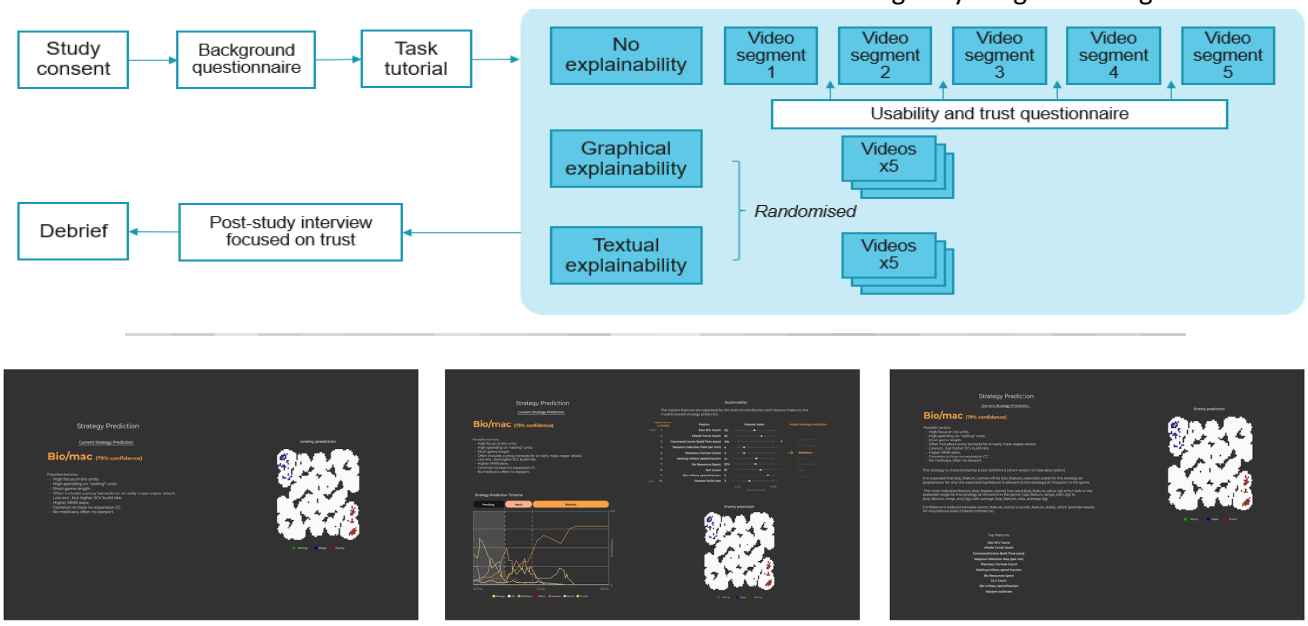

Figure 15. User Questionnaires

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use this UI to play StarCraft II	○	○	○	○	○
I found the UI unnecessarily complex	○	○	○	○	○
I thought the UI would be easy to use when playing StarCraft II	○	○	○	○	○
I think that I would need the support of a technical person to be able to use the UI	○	○	○	○	○
I found the various functions in the UI were well integrated	○	○	○	○	○
I thought there was too much inconsistency in this UI	○	○	○	○	○
I would imagine that most people would learn to use this UI very quickly	○	○	○	○	○
I would find the UI very cumbersome to use	○	○	○	○	○
I would feel very confident using the UI	○	○	○	○	○
I would need to learn a lot of things before I would get going with this UI	○	○	○	○	○

13.1 Interview questions

General feedback

- Among the 3 UIs you saw today, which was your most preferred? Why?
- How could your most preferred UI be improved, e.g. by combining it all in a single UI?

Deeper dive into trust given 2 levels of accuracy data

- Overall, "From a scale of 1-5, what was your feeling of trust in the model?" Please explain the reasoning for your given rating.
- I'll now give you some additional information on the model's accuracy. Throughout the game, it was approximately 88% (or 68%) accurate in predicting the right strategy. How does this information change your perception of the model change?

	No, not at all	No	Not sure	Yes	Yes, completely
Do you have a feeling of trust in the UI?	○	○	○	○	○
According to you, are the outputs of the UI predictable?	○	○	○	○	○
Do you think the UI is a reliable source of gameplay information?	○	○	○	○	○
According to you, is the UI efficient for playing StarCraft II?	○	○	○	○	○

- Unsupervised approach to get clusters ◊ Labelled by hand using domain experts (ground truth) ◊ Fed into classifier (this is what accuracy refers to)
- Finally, "From a scale of 1-5, what was your feeling of trust in the model?" Please explain the reasoning for your given rating.

Transferability to operational planning

- Now, I want you to think outside of StarCraft II about the transferability of the model. Imagine you have access to a similar model for evacuation of non-military personnel from a dangerous evolving situation.
- Would you trust a model like this to help you make decisions in [scenario]? Why or why not?
- What would you want to see in a User Interface to help you make decisions around [scenario]?