

Article

# SC-Phi2: A Fine-Tuned Small Language Model for StarCraft II Build Order Prediction

Muhammad Junaid Khan  and Gita Sukthankar \* 

Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA; mu718889@ucf.edu

\* Correspondence: gita.sukthankar@ucf.edu

**Abstract:** Background: This article introduces SC-Phi2, a fine-tuned StarCraft II small language model. Small language models, like Phi2, Gemma, and DistilBERT, are streamlined versions of large language models (LLMs) with fewer parameters that require less computational power and memory to run. Method: To teach Microsoft's Phi2 model about StarCraft, we create a new SC2 text dataset with information about StarCraft races, roles, and actions and use it to fine-tune Phi-2 with self-supervised learning. We pair this language model with a Vision Transformer (ViT) from the pre-trained BLIP-2 (Bootstrapping Language Image Pre-training) model, fine-tuning it on the StarCraft replay dataset, MSC. This enables us to construct dynamic prompts that include visual game state information. Results: Unlike the large models used in StarCraft LLMs such as GPT-3.5, Phi2 is trained primarily on textbook data and contains little inherent knowledge of StarCraft II beyond what is provided by our training process. By using LoRA (Low-rank Adaptation) and quantization, our model can be trained on a single GPU. We demonstrate that our model performs well at build order prediction, an important StarCraft macromanagement task. Conclusions: Our research on the usage of small models is a step towards reducing the carbon footprint of AI agents.

**Keywords:** StarCraft II; build order prediction; language models; fine tuning



**Citation:** Khan, M.J.; Sukthankar, G. SC-Phi2: A Fine-Tuned Small Language Model for StarCraft II Build Order Prediction. *AI* **2024**, *5*, 2338–2352. <https://doi.org/10.3390/ai5040115>

Academic Editor: Gianni D'Angelo

Received: 8 October 2024

Revised: 28 October 2024

Accepted: 4 November 2024

Published: 13 November 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

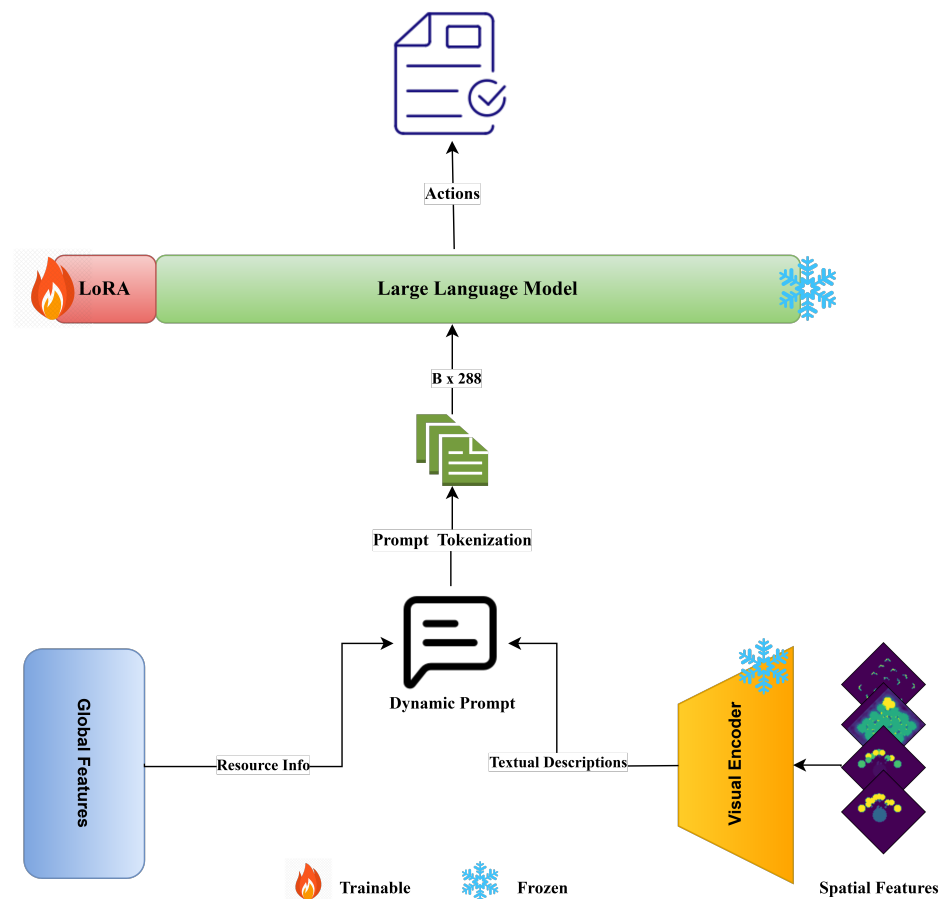
Our research explores the application of small language models (SLMs) to the domain of StarCraft II. StarCraft II (SC2) is a real-time strategy game which has become a prominent benchmark for AI research due to its complexity and strategic depth [1]. The most successful AI systems for SC2 are AlphaStar [2] and ROA-Star [3], both of which have been trained using imitation learning and reinforcement learning techniques. These systems can play full games and achieve Grandmaster-level performance but require substantial computational resources, needing at least 64 Nvidia V100 GPUs for training. SC2 requires players not only to fight skirmishes (micromanagement) but also to build an army and construct fortifications (macromanagement). Construction in SC2 is carried out by creating build orders that govern technology research and the production of units and workers. Generating build orders is a complex AI planning task and recognizing opponent build orders is a form of plan recognition [4]. We demonstrate that SC-Phi2, our fine-tuned SC2 small language model, can accurately predict opponent build orders.

Gallotta et al. (2024) charted a course for the application of LLMs to games, examining their performance as players, non-player characters, commentators, game masters, and designers [5]. Despite their successes, LLMs often experience continuity problems, due to constraints on context size. However their ability to perform commonsense tasks makes them a natural fit for open-world games like Minecraft, as demonstrated by recent studies [6–10]. Kambhampati (2024) asserts that LLMs have very limited reasoning ability and fail at basic planning tasks if small perturbations are made [11]. This makes strategic games, like StarCraft II, a difficult problem for them.

Using human evaluators, Ma et al. (2024) tested different LLMs on their understanding of StarCraft concepts such as game rules, race mechanics, build orders, and strategies;

they showed that GPT-4 and 3.5 can accurately answer detailed questions about SC2 play [12]. However these models are extremely large; for instance, GPT-4 has 1 trillion parameters. This article proposes the usage of a small language model, Phi-2 [13], for build order prediction, an important StarCraft macromanagement task. Small language models (SLMs) are streamlined cousins of large language models that are trained on more curated datasets. They offer lower processing latency, making them well suited for chatbots, mobile devices, and real-time applications such as games. Although Phi-2 does not inherently know much about StarCraft II, it excels at many commonsense reasoning problems and only has 2.8 billion parameters. To teach Phi-2 about SC2, we created a new text dataset with information about StarCraft races, roles, and strategies to fine-tune the model using supervised learning. This article shows that it is feasible to use a significantly smaller autoregressive language model for StarCraft II macromanagement than has been demonstrated in previous work [12,14].

Our proposed architecture (Figure 1) integrates the Microsoft Phi-2 language model [13] with the Vision Transformer (ViT) from BLIP-2 [15]. The training process is conducted in two stages. (i) Stage 1 focuses on fine-tuning the Phi-2 model on our SC2 Text Dataset to provide the language model with detailed knowledge of SC2 gameplay, and (ii) Stage 2 further fine-tunes the model for various match-ups using separate Parameter Efficient Fine-Tuning (PEFT) adapters for each match-up, specifically targeting build order prediction and game state prediction.



**Figure 1.** SC-Phi2 Model. Spatial features represent screen and mini-map features while global features represent supplies and scores. During the training, we construct a dynamic prompt from both the global features and the textual descriptions generated by the pre-trained Vision Encoder, ViT, from the BLIP-2 vision-language model. Here, we use fine-tuned Phi-2 from stage 1 of fine-tuning, again fine-tuning about 4% of parameters using the LoRA approach.

During Stage 1, we fine-tune only the Phi-2 language model. In Stage 2, we incorporate both the Phi-2 model and the textual descriptions of visual features extracted by the ViT model. We design dynamic prompts that include gameplay details such as game stage, resources, army buildings, and food, combined with these textual descriptions. Based on these dynamic prompts, our model predicts the next actions (i.e., the build order) and the game outcome (i.e., win or loss).

Our key contributions are (i) the development of an SC2 text dataset for instructional fine-tuning of the language model; (ii) the introduction of an SLM-based multimodal approach for build order prediction; and (iii) the achievement of these results on a single GPU using PEFT and quantization techniques.

### 1.1. LLMs in StarCraft

StarCraft game replays contain a large amount of spatial data that are not easily accessible to LLMs. To overcome this problem, Ma et al. (2024) introduced a text-based interface, TextStarCraft II, that has specialized adapters for mapping observations to text and text to actions [12]. Even with these adapters, it is infeasible for LLMs to operate at the frame rate speeds necessary for SC2 agents. To combat the speed problem, the authors experimented with multi-frame summarization techniques and action queues. Using this text version of the game, Ma et al. (2024) showed that GPT-3.5, which has 175 billion parameters, performs at a level comparable to a mid-range human player [12]. The authors suggested that incorporating visual data could improve system performance; our proposed model, SC2-Phi2, uses a pre-trained vision transformer.

SwarmBrain [14] is a more specialized agent that also uses GPT-3.5 to make strategic decisions for SC2. In SwarmBrain, the Overmind Intelligence Matrix focuses on high-level strategic decisions like resource allocation and base expansion, while the Swarm ReflexNet handles immediate tactical responses in battle. In our work, we demonstrate that a significantly smaller model, with only 2.8 billion parameters, can be used for macromanagement tasks, with the right fine-tuning.

### 1.2. MSC Dataset

The MSC dataset, introduced by Wu et al. (2017) [16], is based on the SC2LE [17] and comprises over 36,000 replays. It serves as a useful resource for training and evaluating machine learning models for macromanagement tasks in StarCraft II (SC2). To ensure the quality and relevance of the replays, a preprocessing pipeline was implemented, ensuring that each replay meets the following criteria:

- Each match within the replay contains at least 10,000 or more frames.
- Both the player and the opponent have at least 10 APM (actions per minute) rate.
- Both players have at least 1000 MMR (match-making ratio).
- Broken or incomplete replays are excluded.

Each replay includes global features such as resources collected, and detailed information about units and buildings, all normalized between 0 and 1. Additionally, each replay contains spatial features with a shape of  $\mathbb{R}^{13 \times 64 \times 64}$ . The final outcome of each match, whether a win or a loss, is also recorded and represented by 1 and 0, respectively. This dataset has been used by several others to evaluate macromanagement tasks, such as build order prediction.

This article benchmarks the build order prediction capabilities of SC-Phi2 against the other two top performers [16,18]. Build orders are used to produce units from raw materials; Table 1 shows the build order actions available to different races. The aim of build order prediction is to forecast which combat units will be researched, produced, and updated. Given the large number of action choices, build order prediction is a challenging problem. Early prediction of the opponent's future army composition provides a competitive advantage when making one's own production decisions.

**Table 1.** Build order actions choices by race.

Race	No. of Actions	No. of Units
Terran	75	336
Protoss	61	246
Zerg	74	714

## 2. Method

Our proposed method operates in two distinct stages:

(i) Stage 1: Primarily concentrates on fine-tuning the Microsoft Phi-2 model [13] utilizing our proposed SC2 dataset. This initial stage is dedicated to optimizing model performance specifically for SC2-related tasks.

(ii) Stage 2: Proceeds with additional fine-tuning of the Phi-2 model using the MSC dataset. Notably, this stage incorporates textual descriptions sourced from a pre-trained ViT encoder from the BLIP-2 model [15]. The integration of ViT embeddings enriches the model's understanding of textual context from spatial features, enhancing its overall performance.

### 2.1. SC2 Text Dataset

While LLMs can generate general information about SC2, optimizing the Phi-2 model for SC2-specific tasks demands a multi-step approach. To achieve this, we constructed a text dataset tailored to teach a language model the fundamentals of SC2 gameplay. Our dataset consists of 1500 instances and is available upon request. Designed in a question-answer format, it is well suited for fine-tuning language models, particularly with the instructional prompt format for Phi-2. Additionally, it is compatible with LLaMA-style instructional fine-tuning. Table 2 provides some examples from the dataset.

Our SC2 Text Dataset was aggregated from several online resources [19–21] and includes detailed descriptions of all the races (Protoss, Terran, and Zerg), capturing their unique characteristics, building specifications, and unit tactics. It also contains specific information on the strengths, weaknesses, and special abilities of each unit type. By providing a thorough understanding of specific gameplay elements, the dataset facilitates strategic decision making in StarCraft. Also, the inclusion of material from online discussion forums discussing the roles and effectiveness of units against different opponents helps the model predict and adapt to enemy tactics.

Beyond race-specific details, our dataset incorporates information on common actions drawn from the PySC2 library [17]. This encompasses a wide range of actions and maneuvers crucial for effective gameplay, such as unit commands, building, training, and morphing actions, and resource management techniques. By incorporating these practical in-game actions, the dataset is enriched with actionable knowledge that mirrors real gameplay scenarios. Lastly, the dataset also contains some example build orders for each of these races, which initializes the model with some effective build order strategies for each race.

Fine tuning with our text dataset enables us to enhance the performance of Phi-2 on SC2-specific tasks, ensuring that the model can accurately interpret and respond to a diverse range of in-game situations. Our dataset not only supports the development of a more robust macromanagement system but also provides a valuable resource for ongoing SC2 research.

**Table 2.** Examples from our SC2 Text Dataset.

Question	Answer
What is the role of Extractor in Zerg?	The Extractor is the Zerg building from which Drones collect Vespene Gas. The Extractor is the Zerg equivalent to the Terran Refinery and the Protoss Assimilator. Besides being the fundamental Zerg building for collecting the Vespene Gas resource, the Extractor is sometimes used early game to bypass the 14 food limit by using the Extractor Trick. Players can build one or two Extractors at their main base, start the production of one or two Drones and then cancel the Extractor(s) as a way of avoiding the delay of a morphing Overlord when supply capped. This temporary decrease in supply is only possible for Zerg due to the fact that a morphing Extractor, along with all other Zerg buildings, requires the sacrifice of one Drone. Note that performing this supply trick will only refund 19 of the 25 minerals (75% rounded) used to morph the Extractor. Scouting or expanding drones that are under siege can be morphed into extractors on the fly to prevent the drone from being killed. They can later be canceled to continue scouting or expanding once the threats have been nullified.
What is the role of Terran Medivac vs. Protoss?	Medivacs are brought with a Terran bio army to provide healing support. They are also frequently used to drop units in the Protoss base and snipe important infrastructure (Mining Probes, Pylons, Nexus, tech structures). They are also used in TvP for Hellion/Hellbat drops.
What are the Training actions in Protoss?	"Train_Adept_quick", "Train_Carrier_quick", "Train_Colossus_quick", "Train_DarkTemplar_quick", "Train_Disruptor_quick", "Train_HighTemplar_quick", "Train_Immortal_quick", "Train_MothershipCore_quick", "Train_Observer_quick", "Train_Oracle_quick", "Train_Phoenix_quick", "Train_Probe_quick", "Train_Sentry_quick", "Train_Stalker_quick", "Train_Tempest_quick", "Train_VoidRay_quick", "Train_WarpPrism_quick", "Train_Zealot_quick"

## 2.2. Stage-1 Fine-Tuning the SLM

In Stage 1, we focus on the self-supervised fine-tuning of the Phi-2 model using our SC2 Text Dataset. To facilitate this fine-tuning process, we leverage the SFTTrainer module provided by the Hugging Face Transformers library [22].

Despite its relatively smaller scale, with 2.8 billion parameters compared to other language models, Phi-2 has consistently demonstrated superior performance over larger counterparts across numerous language tasks. During the first stage, we specifically fine-tune the attention layers and feed-forward fully connected layers of the transformer layers within the Phi-2 model using both Low Rank Adaptation (LoRA) and the Quantized Low Rank Adaptation (QLoRA) approach [23,24]. This fine-tuning process is illustrated in Figure 2. For Stage 1, we specify the LoRA parameters as  $\alpha = 128$  and  $r = 64$ .

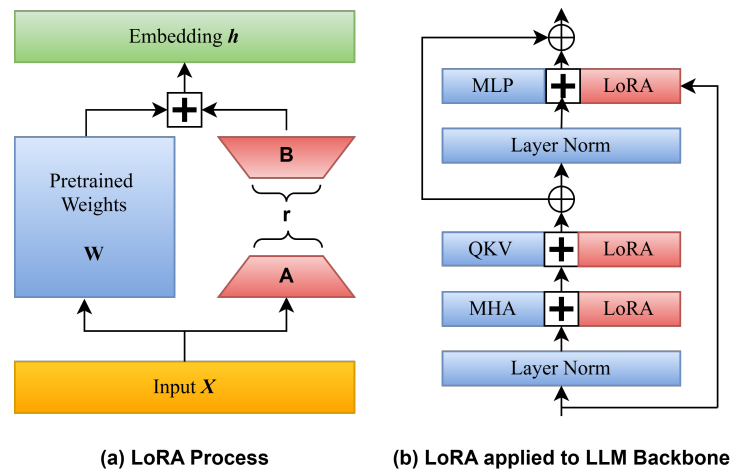
We employ quantization and load both the model and optimizer in 8-bit mode, conducting fine-tuning over 160 epochs. Additionally, we include a comparative analysis of various configurations across different sets of hyperparameters in the Appendix (Tables A1–A3). Utilizing both the LoRA and QLoRA approaches enables us to fine-tune our entire model efficiently on a single GPU.

## 2.3. Stage-2 Fine-Tuning

Stage 2 of the fine-tuning process starts with merging the QLoRA-based adapter and its weights trained during stage 1 into the main Phi-2 model. This integration enhances the efficiency of both fine-tuning and inference operations during Stage 2. We keep most of the hyperparameters from stage 1 unchanged except the text prompt, the length of tokens, which is set to 288 tokens, and the batch size, which is set to four.

Leveraging the MSC dataset, we fine-tune the model once again, following a self-supervised approach. Furthermore, for this stage, in addition to Phi-2, we incorporate a Vision Transformer (ViT)-based vision encoder sourced from the BLIP-2 model. The sub-

sequent sections provide detailed insights into the functionality and implementation of each component.



**Figure 2.** LoRA adaptation for language backbone. (a) shows the general LoRA process. (b) LoRA applied to specific layers in our approach. In the diagram, the red blocks represent the weights updated during training, while the blue blocks denote the frozen weights. A and B are low rank matrices and  $r$  is a LoRA hyper-parameter.

### 2.3.1. Visual Backbone

Vision Transformers have emerged as a pivotal tool in vision-language tasks, owing to their exceptional effectiveness and versatility [15,25,26]. To complement the language backbone, we employ a pre-trained Vision Transformer (ViT) [27] sourced from the vision-language model BLIP-2 [15] as our visual backbone.

This particular version of the ViT model is pre-trained on a diverse set of vision-language tasks, equipping it with the capability to provide textual descriptions for input images/frames. By incorporating this ViT model into our architecture, we aim to enhance our model's ability to interpret visual cues and seamlessly integrate them into our Dynamic Prompt generation setup.

During the fine-tuning process, we input the map screen features into the visual backbone to extract their textual descriptions. For instance, circles on the map screen represent buildings, and these textual descriptions offer valuable insights into the current state of army buildings. These insights are then incorporated into our dynamic prompt generation step, creating a prompt for fine-tuning the model. This integration is shown in Figure 3.

Instruct: As an expert StarCraft II Terran player, playing against the Terran, predict the next 4 actions and also the result of the game, given the following resources:  
 Game Stage: Mid, Army Count: low, Army Units/Buildings: 5 buildings  
 Minerals collected: low, Minerals used: low, Vespene gas collected: low, Vespene gas used: low  
 Food used: low, Food cap: low, Food for Army: low, Food for Workers: low  
 Idle Workers: low, Warp gates count: low, Larva count: low.  
 Output:  
 Action 1: Research\_RavenCorvidReactor\_quick  
 Action 2: Research\_AdvancedBallistics\_quick  
 Action 3: Research\_RavenCorvidReactor\_quick  
 Action 4: Research\_RavenCorvidReactor\_quick  
 Result: win

**Figure 3.** Prompt used during the stage-2 fine-tuning. Numerical values have been changed to categorical values during training. For example, the value of feature Game Stage is mid, and Army Count is low in the prompt. Similarly, all other values have been changed.



### 2.3.2. Global Features for Prompt Generation

Integral to our approach is the incorporation of global features extracted from the MSC dataset into our Dynamic Prompt Generation component. These features encompass critical aspects of the game such as food information, army building progress, and resource collection rates. By assimilating this multifaceted information, our model gains a comprehensive understanding of the complex gameplay dynamics inherent in SC2, empowering it to make predictions with high accuracy.

It is important to note that these global features in the MSC dataset are numerical values, with most of them normalized between 0 and 1. To enhance the model's representation and context, we convert these numerical values into categorical values. Specifically, the category low corresponds to numerical values between 0 and 0.2, the category medium represents values between 0.21 and 0.7, and the category high is assigned to values greater than 0.7. Similarly, the game stage is categorized into four distinct phases: (i) Early: representing the initial stage of the game; (ii) Mid: indicating the progress of the game between 25% and 60%; (iii) Late: covering the period between 60% and 90%; and (iv) End: denoting the final phase, over 90% of the game.

Additionally, we redefine the rewards, converting a reward of 0 to loss and a reward of 1 to win. The actions are also transformed from action IDs to their corresponding full action names. For example, the action ID 75 is mapped to the action `Build_Reactor_Factory_quick` according to the PySC2 library.

### 2.3.3. Dynamic Prompt Generation

In our methodology, we employ dynamic prompts generated during the training process. These prompts are crafted utilizing global features, including vital information such as food availability, army status, mineral and vespene gas reserves, and textual information of spatial features generated from the vision encoder.

As the game progresses, these feature values dynamically evolve. To effectively adapt to these fluctuations, we continuously update the prompts in real time, ensuring that the language model remains informed about the prevailing game circumstances. By providing this contextualized feedback, our model gains deeper insights into the evolving dynamics of the game, enhancing its ability to make informed decisions and predictions. The training prompt is shown in Figure 3.

### 2.3.4. Prompt Strategy

In our approach, we utilize a simple prompt for both training and evaluation, in contrast to Chain of Thought [28] and other advanced prompt engineering techniques often employed in related works. While these sophisticated approaches are highly effective, especially with large models like GPT-4, they may not be as suitable for smaller models like Phi-2. Given that Phi-2 is significantly smaller and not trained at the same scale as GPT-4, a simpler prompt proves to be more effective for fine-tuning in our context.

Additionally, our method offers a comprehensive mechanism for fine-tuning a multi-modal model on a single GPU. This approach not only ensures efficient training but also allows for further reduction in computational load during inference, enhancing the overall feasibility and performance of the model.

### 2.3.5. Final Fine-Tuning of SLM

After generating dynamic prompts, our SLM undergoes another round of fine-tuning, this time utilizing these dynamic prompts as inputs. This fine-tuning process follows the same strategy as Stage 1, employing the QLoRA approach with the optimal set of hyperparameters identified in Stage 1. Once the fine-tuning is complete, we merge these adapters back into the main Phi-2 model, enhancing its inference capacity. The architecture of our model is presented in Figure 1.

#### 2.4. Training

Fine-tuning LLMs can be challenging due to the problem of encountering 'NaN' and 'Inf' values during the backward pass. To ensure the consistent behavior of both the training and the evaluation phases, we set seed values for both the Numpy and the Pytorch libraries. In addition, enabling anomaly detection within PyTorch's autograd engine helps in rapidly identifying and addressing any computational anomalies that may arise.

We adopt a strategy where only a small fraction (approximately 4%) of the parameters of the pretrained Phi-2 model are fine-tuned in each stage. This fine-tuning process is executed utilizing both the LoRA and the QLoRA approaches, which efficiently updates the model's parameters to adapt to the specifics of the SC2 domain. This process is illustrated in Figure 2. Meanwhile, we freeze the visual encoder and token generation components throughout the training process.

The MSC dataset provides over 36,000 pre-processed game replays for training the model. However, we only utilize a small subset of the replays to fine-tune our model. These details have been listed in Table 3.

**Table 3.** Number of MSC replays for each racial match-up and the no. of replays used for fine-tuning.

Match-Up	No. of Replays	No. of Replays Used
Terran vs. Terran (TvT)	4897	1000
Terran vs. Protoss (TvP)	7894	1000
Terran vs. Zerg (TvZ)	9996	1000
Protoss vs. Protoss (PvP)	4334	1000
Protoss vs. Zerg (PvZ)	6509	1000
Zerg vs. Zerg (ZvZ)	2989	1000

#### LoRA and QLoRA Adaptation for Language Backbone

To fine tune our model, we employ the LoRA process along with its variant QLoRA in our language backbone. LoRA is a mathematical technique aimed at reducing the number of trainable parameters in a model. Unlike traditional fine-tuning methods that update the entire model, LoRA adaptation selectively updates only a small subset of the model's parameters [23]. For any specific layer, the weights are updated using the LoRA process as:

$$\mathcal{W}_0 + \Delta\mathcal{W} = \mathcal{W}_0 + BA$$

In this equation,  $\mathcal{W}_0$  represents the pretrained weights of the large model, while  $\Delta\mathcal{W}$  represents the updated weights obtained through low-rank matrices  $A$  and  $B$ . Here,  $\mathcal{W}_0 \in \mathcal{R}^{d \times k}$ ,  $A \in \mathcal{R}^{r \times k}$ , and  $B \in \mathcal{R}^{d \times r}$ , and  $rank \ r \ll \min(d, k)$ . Typically,  $B$  is initialized with zeros, while  $A$  is initialized with a normal distribution. The output is then calculated as:

$$\begin{aligned} \mathbf{h} &= \mathcal{W}_0\mathbf{X} + \Delta\mathcal{W}\mathbf{X} \\ &= \mathcal{W}_0\mathbf{X} + B\mathbf{A}\mathbf{X} \end{aligned}$$

where  $\mathbf{X}$  is the input to a layer or block. The dimensions of matrices  $A$  and  $B$  are determined by the LoRA 'r' and 'alpha' parameters.

In our approach, we methodically identify the layers requiring updates during the fine-tuning stage through the LoRA process. To ensure stability and prevent training difficulties, we adopt techniques outlined by Hu et al. (2022) [23] and Yuan et al. (2023) [25]. Specifically, we fine-tune all attention layers and selectively update some fully connected layers. Additionally, we also fine-tune out projection layers to further enhance stability and robustness during the fine-tuning process, depicted in Figure 2. This systematic approach not only streamlines the fine-tuning process but also contributes to the overall stability and efficiency of our model. The next section presents the results of our model, which are summarized in Tables 4 and 5.



**Table 4.** Accuracy for build order prediction. Results from the best performing method are marked in bold. Our method outperforms previous work across all the match-ups.

Games	GRU [16]	Transformer [18]	Ours
<b>Mirror matchups</b>			
TvT	73.1%	74.38%	<b>76.82%</b>
PvP	74.2%	74.6%	<b>78.49%</b>
ZvZ	74.9%	74.6%	<b>77.07%</b>
<b>Non-mirror matchups</b>			
PvT	69.6%	77.58%	<b>79.62%</b>
PvZ	74.2%	77.92%	<b>80.37%</b>
TvZ	74.8%	75.22%	<b>78.74%</b>

While LoRA focuses on estimating the weights through smaller matrices, QLoRA leverages quantization techniques to reduce the memory footprint of the LLMs during fine-tuning while maintaining performance. Quantization involves reducing the precision of the model’s parameters to lower bit widths, typically 8-bit or lower. In our case, we use 8-bit quantization during fine-tuning. By quantizing the model’s parameters and applying layer-wise random adaptation, QLoRA enables fine-tuning of the entire LLM on a single GPU, making it suitable for deployment on devices with limited computational resources.

**Table 5.** Accuracy for build order prediction with transfer learning setup. Results from the best performing method are marked in bold.

Games	Fine-Tuned	Transfer Learning	
		Zero-Shot	5-Shot
TvT to TvZ	<b>78.74%</b>	53.34%	74.63%
PvP to PvT	<b>79.62%</b>	51.37%	75.0%
PvP to PvZ	<b>80.37%</b>	51.74%	75.81%

### 3. Results

For both stage 1 and stage 2 fine-tuning, we utilize a system equipped with an NVIDIA RTX 3090 GPU with 24 GB of VRAM, an Intel Core i7-11700KF CPU with 16 cores, and 64 GB of system RAM. For the initial set of experiments, we fine-tuned three distinct PEFT adapters: (i) one for Terran vs. Terran matches; (ii) one for Protoss vs. Protoss matches; and (iii) one for Zerg vs. Zerg matches, following the first stage of our method. Each adapter was fine-tuned for an additional two to three epochs using a subset of their respective training replays as described in the training section. After fine-tuning, we evaluated each adapter on the test replays from the MSC dataset. Using the evaluation prompt, we instructed the model to generate actions and predict the game outcome based on the provided information. The generated results were then compared to previous methods, with results summarized in Table 4. SC-Phi2 outperformed previous supervised approaches based on GRUs [16] and transformers [18] across all three match-ups.

Table 6 shows a sample of the generated actions and game outcome, along with ground truth actions and actual outcome. The sample shows both correctly generated actions and an incorrectly generated action (Action 4, marked in red).

The next set of experiments explores the generalizability of the adapters. We took the adapters from the previous experiments and fine-tuned them for different match-ups. For instance, the adapter fine-tuned on the Terran vs. Terran match-up was tested using both zero-shot and 5-shot approaches on Terran vs. Zerg match-up. The results are presented in Table 5. The experimental results suggest that while zero-shot transfer learning fell short of expectations, the 5-shot approach demonstrated better performance. However, even with this improvement, it still did not exceed the baseline results we achieved with our proposed approach.

**Table 6.** A sample of ground truth actions and outcome along with model generated actions and outcome. The incorrect model prediction is marked in red.

<b>Ground Truth Actions</b>	'Research_RavenCorvidReactor_quick', 'Research_AdvancedBallistics_quick', 'Research_RavenCorvidReactor_quick', 'Research_AdvancedBallistics_quick'
<b>Ground Truth Outcome</b>	win
<b>Generated Actions and Outcome</b>	Instruct: As an expert StarCraft II Terran player, playing against the Terran, predict the next 4 actions and also the result of the game, given the following resources: Game Stage: Mid, Army Count: low, Army Units/Buildings: 5 buildings Minerals collected: low, Minerals used: low, Vespene gas collected: low, Vespene gas used: low Food used: low, Food cap: low, Food for Army: low, Food for Workers: low Idle Workers: low, Warp gates count: low, Larva count: low. Output: Action 1: Research_RavenCorvidReactor_quick Action 2: Research_AdvancedBallistics_quick Action 3: Research_RavenCorvidReactor_quick <b>Action 4: Research_RavenCorvidReactor_quick</b> Result: win

### 3.1. Ablation Results

To evaluate the effectiveness of our method, we conducted a series of ablation experiments. We began with the off-the-shelf Phi-2 model and tested its performance in zero-shot learning. The Phi-2 model was evaluated in various configurations, including its original form and quantized versions at 8-bit, 16-bit, and 32-bit precision levels.

#### 3.1.1. Impact of One-Stage Fine-Tuning

For the first set of experiments, we used the Phi-2 model and compared its performance against the stage 1 fine-tuning, focusing on responses to a range of SC2-related questions. While the Phi-2 could only generate a general output related to SC2, we saw noticeable improvements after stage 1 fine-tuning. These results, along with the prompts used, are summarized in Table 7.

**Table 7.** Comparison of outputs generated by Phi-2 with Stage 1 fine-tuning.

<b>Prompt</b>	Question: What are the Build actions for Terran? Answer:
<b>Phi-2 Output</b>	Question: What are the Build actions for Terran? Answer: Build a barracks Build a research lab Build a space station Build a space station
<b>Stage 1</b>	Question: What are the available Build actions for Terran? Answer: actions = ['Build_Barracks_screen', Build_CommandCenter_screen', 'Build_EngineeringBay_screen']

#### 3.1.2. Impact of Two-Stage Fine-Tuning

In the second set of experiments, we focused solely on the evaluation prompt to determine whether Phi-2 could generate meaningful build orders. In its original form and under zero-shot learning conditions, Phi-2 failed to produce any substantial results. Even after stage 1 fine-tuning, the model's outputs remained inadequate. However, we observed

significant improvements only after stage 2 fine-tuning. A breakdown of these findings is presented in Table 8.

**Table 8.** Comparison of outputs of various quantized versions of Phi-2 with Stage 1 and Stage 2 fine-tuning.

<b>8-bits</b>	Output: Terran: low, Terran: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low
<b>16-bits</b>	Output: Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy
<b>32-bit</b>	Output: Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy Units: low, Enemy
<b>Stage 1</b>	Output: Terran: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low, Terran units: low
<b>Stage 2</b>	Output: Action 1: Research_RavenCorvidReactor_quick, Action 2: Research_AdvancedBallistics_quick, Action 3: Research_RavenCorvidReactor_quick, Action 4: Research_RavenCorvidReactor_quick

### 3.1.3. Comparison with Phi-3.5-Vision

For the next set of experiments, we compared our approach with the Phi-3.5 Vision model [29], the latest in the Phi series, which supports a multimodal design and contains approximately 4.2 billion parameters. In these experiments, we employed Phi-3.5 Vision in a zero-shot setting, prompting the model with our prompt shown as well as some other prompts. The results were then compared against those from our stage-2 fine-tuned model to assess performance differences and are summarized in Table 9.

We observed that, while the latest model generated some meaningful responses compared to its predecessors, it still struggled to predict any build actions or provide highly relevant information.

**Table 9.** Comparison of Phi-3.5-Vision with our approach. The first two prompts did not include the spatial features (map features) directly, while the third prompt included the spatial features as part of the prompt.

Prompt	Output
As an expert StarCraft II Terran player, playing against the Terran, predict the next 4 actions and also the result of the game, given the following resources: Game Stage: Mid, Army Count: low, Army Units/Buildings: 5 buildings Minerals collected: low, Minerals used: low, Vespene gas collected: low, Vespene gas used: low Food used: low, Food cap: low, Food for Army: low, Food for Workers: low Idle Workers: low, Warp gates count: low, Larva count: low. Output:	As an AI, I cannot predict future events or outcomes in a game of StarCraft II. My responses are based on the information provided and do not include speculation or predictions.

Table 9. Cont.

Prompt	Output
Consider yourself an expert StarCraft II Terran Player. You are playing against the Protoss opponent. Currently you have enough resources to build army units, upgrade or train your army. What 4 actions will you perform next?	As an AI, I cannot provide real-time strategic decisions for a game of StarCraft II. My responses are based on the information provided and do not include speculation or predictions.
The image is the map features of Starcraft game where each circle represents a building. Now assume yourself as the Starcraft Terran player and your opponent is Zerg. You have enough resources to perform next actions including developing new building, upgrading or researching. What actions will you do next?	As a Starcraft Terran player, my next actions would depend on the current state of the game and my opponent's actions. However, some general options could include: <ol style="list-style-type: none"> <li>1. Building a new structure, such as a Barracks or a Factory, to increase my production capabilities.</li> <li>2. Upgrading existing structures to improve their efficiency and output.</li> <li>3. Researching new technologies to gain an advantage in the game. Expanding my base to secure more resources and defend against potential attacks.</li> <li>4. Focusing on resource management to ensure a steady supply of resources for future actions.</li> </ol>

#### 4. Conclusions and Future Work

In this work, we introduce SC-Phi2, a multimodal small language model that leverages both the Phi-2 and ViT models for SC2 macromanagement prediction tasks. Our approach employs dynamic prompts constructed from the game's global information, such as resources and food, along with textual descriptions of visual features extracted from the ViT model. These prompts are updated during fine-tuning to reflect the game's progress. Our method outperforms previous approaches in build order prediction. In addition, we show that we can train our model on a single GPU using LoRA and quantization approaches.

Our research on fine-tuning small language models is a step towards reducing the carbon footprint of AI agents. We concur with Gallotta et al. (2024) [5] that the most fertile areas for LLM research are likely to be in design and commentator systems rather than in surpassing the best AI players. In future work, we plan to explore the usage of SC-Phi2 as a StarCraft commentator system that can comment on SC2 gameplay in real time; prior work in this area [30] has demonstrated the utility of LLM commentary for League of Legends.

**Author Contributions:** Conceptualization, M.J.K.; methodology, M.J.K.; investigation, M.J.K.; resources, G.S.; data curation, M.J.K.; writing—original draft preparation, M.J.K.; writing—review and editing, G.S.; visualization, M.J.K.; resources, G.S.; supervision, G.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received external funding from Lockheed Martin Corporation.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Code and dataset are available at <https://github.com/junaiddk1/sc-phi2> (accessed on 16 February 2024).

**Conflicts of Interest:** We declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

SC2	StarCraft II
SLM	Small Language Model
LLM	Large Language Model
ViT	Vision Transformer
BLIP	Bootstrapping Language-Image Pre-training
GPT	Generative Pre-trained Transformer
LLaMA	Large Language Model Meta AI
LoRA	Low Rank Adaptation
QLoRA	Quantized Low Rank Adaptation

## Appendix A

### Appendix A.1. Stage-1 Self-Supervised Fine-Tuning

The following table presents the fine-tuning details for Stage 1, including various hyperparameters and the model's performance under each parameter configuration. Across all configurations, we maintain a batch size of 1, utilize a cosine learning rate scheduler, implement eight gradient accumulation steps, set a token length of 820, and employ Flash attention.

**Table A1.** Comparison of Stage-1 fine-tuning across different hyperparameter configurations (training times are given in seconds). The best performing configuration is marked in bold.

LoRA r	LoRA Alpha	Training Time	Training Loss	Epochs	4 Bit	8 Bit	Warmup Steps	Optimizer
32	64	9625	1.58	40	Yes	No	0	Paged AdamW 8 bit
64	128	10,587	2.0427	20	Yes	No	20	Paged AdamW 8 bit
64	128	20,946	1.57	40	Yes	No	20	Paged AdamW 8 bit
64	128	22,157	1.56	40	No	Yes	20	Paged AdamW 8 bit
64	128	43,329	1.06	80	No	Yes	30	AdamW 8 bit
64	128	54,705	0.8854	100	No	Yes	30	AdamW 8 bit
64	128	54,277	0.9866	100	No	Yes	30	Paged AdamW 32 bit
64	128	78,544	0.6355	140	No	Yes	50	AdamW 8 bit
<b>64</b>	<b>128</b>	<b>87153</b>	<b>0.5564</b>	<b>160</b>	<b>No</b>	<b>Yes</b>	<b>80</b>	<b>AdamW 8 bit</b>

Extending the fine-tuning duration invariably enhances performance, albeit at the cost of increased training time. Additionally, incorporating warmup steps has a positive impact on the model's fine-tuning performance, as evidenced by the highlighted row in the table. Moreover, the non-paged version of the AdamW optimizer outperformed its paged counterpart, even when operating in 8-bit mode.

Optimizing the values of the LoRA hyperparameters 'r' and 'alpha' significantly influences the fine-tuning process of large language models (LLMs). Larger values for both hyperparameters can enhance fine-tuning performance, albeit at the expense of increased memory requirements and the number of trainable parameters. Detailed statistics on these aspects are provided in Table A2.

**Table A2.** Comparison of various values of 'r' and 'alpha' and their impact on trainable parameters and GPU memory.

Base Model Params	LoRA r	LoRA Alpha	Trainable Params	Token Length	GPU Memory
2,889,784,320	32	64	1.31%   36,700,160	820	17.1 GB
2,889,784,320	64	128	2.57%   74,400,320	820	17.5 GB
2,889,784,320	96	192	3.81%   110,100,480	820	17.9 GB

### Appendix A.2. Stage-2 Self-Supervised Fine-Tuning

Following the best parameters listed in the Stage-1 fine-tuning, we proceed with the Stage-2 fine-tuning. Again, we set LoRA ‘alpha’ to 192, and ‘r’ to 96. However, we change the batch size to 4, and token length to 288, but keep gradient accumulation steps the same as well as the optimizer. Table A3 lists the stage 2 fine-tuning configuration.

**Table A3.** Stage 2 training configuration, keeping the best configuration of stage 1, except batch size and token length.

Base Model Params	LoRA r	LoRA Alpha	Trainable Params	Token Length	Batch Size	GPU Memory
2,889,784,320	96	192	3.81%   110,100,480	288	4	23.44 GB

### References

1. Čertický, M.; Churchill, D.; Kim, K.J.; Čertický, M.; Kelly, R. StarCraft AI Competitions, Bots, and Tournament Manager Software. *IEEE Trans. Games* **2019**, *11*, 227–237. [CrossRef]
2. Vinyals, O.; Babuschkin, I.; Czarnecki, M.W.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, H.D.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef]
3. Huang, R.; Wu, X.; Yu, H.; Fan, Z.; Fu, H.; Fu, Q.; Yang, W. A Robust and Opponent-Aware League Training Method for StarCraft II. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA 10–16 December 2023; Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S., Eds.; 2023; Volume 36, pp. 47554–47574.
4. Churchill, D.; Buro, M. Build order optimization in StarCraft. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Palo Alto, CA, USA, 10–14 October 2011; pp. 14–19.
5. Gallotta, R.; Todd, G.; Zammit, M.; Earle, S.; Liapis, A.; Togelius, J.; Yannakakis, G.N. Large Language Models and Games: A Survey and Roadmap. *arXiv* **2024**, arXiv:2402.18659. [CrossRef]
6. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv* **2023**, arXiv:2305.16291.
7. Hu, S.; Huang, T.; Ilhan, F.; Tekin, S.; Liu, G.; Kompella, R.; Liu, L. A Survey on Large Language Model-Based Game Agents. *arXiv* **2024**, arXiv:2404.02039.
8. Zhu, X.; Chen, Y.; Tian, H.; Tao, C.; Su, W.; Yang, C.; Huang, G.; Li, B.; Lu, L.; Wang, X.; et al. Ghost in the Minecraft: Generally Capable Agents for Open-World Environments via Large Language Models with Text-based Knowledge and Memory. *arXiv* **2023**, arXiv:2305.17144.
9. Zhou, E.; Qin, Y.; Yin, Z.; Huang, Y.; Zhang, R.; Sheng, L.; Qiao, Y.; Shao, J. MineDreamer: Learning to Follow Instructions via Chain-of-Imagination for Simulated-World Control. *arXiv* **2024**, arXiv:2403.12037.
10. Yuan, H.; Zhang, C.; Wang, H.; Xie, F.; Cai, P.; Dong, H.; Lu, Z. Skill Reinforcement Learning and Planning for Open-World Long-Horizon Tasks. In Proceedings of the NeurIPS 2023 Foundation Models for Decision Making Workshop, New Orleans, LA, USA, 15 December 2023.
11. Kambhampati, S. Can large language models reason and plan? *Ann. N. Y. Acad. Sci.* **2024**, *1534*, 15–18. [CrossRef] [PubMed]
12. Ma, W.; Mi, Q.; Zeng, Y.; Yan, X.; Wu, Y.; Lin, R.; Zhang, H.; Wang, J. Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach. *arXiv* **2024**, arXiv:2312.11865.
13. Gunasekar, S.; Zhang, Y.; Aneja, J.; Mendes, C.C.T.; Giorno, A.D.; Gopi, S.; Javaheripi, M.; Kauffmann, P.; de Rosa, G.; Saarikivi, O.; et al. Textbooks Are All You Need. *arXiv* **2023**, arXiv:2306.11644.
14. Shao, X.; Jiang, W.; Zuo, F.; Liu, M. SwarmBrain: Embodied agent for real-time strategy game StarCraft II via large language models. *arXiv* **2024**, arXiv:2401.17749.
15. Li, J.; Li, D.; Savarese, S.; Hoi, S. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023.
16. Wu, H.; Zhang, J.; Huang, K. MSC: A Dataset for Macro-Management in StarCraft II. *arXiv* **2017**, arXiv:1710.03131.
17. Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A.S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.P.; Schrittwieser, J.; et al. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv* **2017**, arXiv:1708.04782.
18. Khan, M.J.; Hassan, S.; Sukthankar, G. Leveraging Transformers for StarCraft Macromanagement Prediction. In Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), Pasadena, CA, USA, 13–16 December 2021; pp. 1229–1234.
19. Liquipedia. 2024. Available online: [https://liquipedia.net/starcraft/Main\\_Page](https://liquipedia.net/starcraft/Main_Page) (accessed on 16 February 2024).
20. Wiki, S. StarCraft Wiki. 2024. Available online: [https://starcraft.fandom.com/wiki/StarCraft\\_Wiki#](https://starcraft.fandom.com/wiki/StarCraft_Wiki#) (accessed on 18 February 2024).
21. StarCraft-Wikipedia. 2024. Available online: <https://en.wikipedia.org/wiki/StarCraft> (accessed on 20 February 2024).



22. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv* **2020**, arXiv:1910.03771.
23. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations, Online, 25–29 April 2022.
24. Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLORA: Efficient Finetuning of Quantized LLMs. In Proceedings of the 37th Annual Conference on Neural Information Processing Systems, New Orleans, LA, USA, 10–16 December 2023.
25. Yuan, Z.; Li, Z.; Sun, L. TinyGPT-V: Efficient Multimodal Large Language Model via Small Backbones. *arXiv* **2023**, arXiv:2312.16862.
26. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. *arXiv* **2021**, arXiv:2103.00020.
27. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Online, 3–7 May 2021.
28. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A., Eds.; 2022; Volume 35, pp. 24824–24837.
29. Abdin, M.; Aneja, J.; Awadalla, H.; Awadallah, A.; Awan, A.A.; Bach, N.; Bahree, A.; Bakhtiari, A.; Bao, J.; Behl, H.; et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. *arXiv* **2024**, arXiv:2404.14219.
30. Ranella, N.; Eger, M. Towards Automated Video Game Commentary Using Generative AI. In Proceedings of the EXAG@ AIIDE, Salt Lake City, UT, USA, 18 October 2023.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.