Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)

Learning to Reason in Round-Based Games: Multi-Task Sequence Generation for Purchasing Decision Making in First-Person Shooters

Yilei Zeng,* Deren Lei,* Beichen Li,* Gangrong Jiang,* Emilio Ferrara, Michael Zyda
University of Southern California
{yilei.zeng, derenlei, beichen, gjiang, emiliofe, zyda}@usc.edu

Abstract

Sequential reasoning is a complex human ability, with extensive previous research focusing on gaming AI in a single continuous game, round-based decision makings extending to a sequence of games remain less explored. Counter-Strike: Global Offensive (CS:GO), as a round-based game with abundant expert demonstrations, provides an excellent environment for multi-player round-based sequential reasoning. In this work, we propose a Sequence Reasoner with Round Attribute Encoder and Multi-Task Decoder to interpret the strategies behind the round-based purchasing decisions. We adopt few-shot learning to sample multiple rounds in a match, and modified model agnostic meta-learning algorithm Reptile for the meta-learning loop. We formulate each round as a multi-task sequence generation problem. Our state representations combine action encoder, team encoder, player features, round attribute encoder, and economy encoders to help our agent learn to reason under this specific multi-player round-based scenario. A complete ablation study and comparison with the greedy approach certify the effectiveness of our model. Our research will open doors for interpretable AI for understanding episodic and long-term purchasing strategies beyond the gaming community.[1]

1 Introduction

Over the last couple of years, we have seen the gaming AI community moving towards training agents in more sophisticated games, like Doom (Kempka et al. 2016), StarCraft (Lin et al. 2017), Minecraft (Guss et al. 2019), and Dota2 (OpenAI et al. 2019). These online games are match-based, fast-paced, highly strategic, and involve adversarial real-time battles.

However, existing studies are restricted to treating the complete continuous game as a single task which would not generalize to the cases of round-based games. We define round-based game as a meta game that can be decomposed into multiple games which can be independent and with different rules. An round-based game reasoning example could be: A professional Dota2 player plays five games in total, but already lose two games in a row. How would he play the third one? Will he choose an aggressive strategy? Such round-based game reasoning is a fundamental problem of building video game AI, as its strategy can also be applied to continuous game cases. For example, the death of a player's character in DOTA2 can be considered as a round or a micro-game which contributes non-equally to the entire meta-game. Utilizing information from a round-based level could be crucial to the complete game reasoning process.

In this work, we are particularly interested in tackling the new challenges of round-based games by proposing a simple scenario: each round only requires one action to receive a result, and each round contributes equally to the entire meta-game. To this end, we introduce a round-based dataset collected from CS:GO professional match replays, which consists of 5167 matches, and each match contains a maximum of 30 rounds and exactly 10 players for two sides. Winning the entire game requires to win at least 16 rounds.

The agent learns the reasoning behind the player's weapon purchasing decisions at the beginning of each round, given the observation of previous rounds' match statistics together with the current round weapon information described in Section 4. We aim to build a human centered AI that learn to reason from human decision makings and in return help interpret the process rather than achieving the best in game performance.

We propose three approaches to deal with such reasoning challenge:

- *Greedy Algorithm*, this model buys the utmost affordable weapon sequence during each round.
- *Sequence Reasoner*, since a player buys a sequence of weapons for each round and each weapon belongs to one of three types, gun, grenade, or equipment, we consider this task as a multi-task sequence generation problem with pre-trained the weapon embedding from the context of weapon sequence.
- *Sequence Reasoner with Round Attribute Encoder*, the model encodes the player's previous round history through Round Attribute Encoder (RAE) into an auxiliary round attribute for the Sequence Reasoner.

* Equal contributions.

[1]Code and data is available at https://github.com/derenlei/CS_Net.

Extensive experiments demonstrate the effectiveness of our proposed third method. However, the result is still not close to the original professional player's level. Thus, we believe that the proposed CS:GO weapon purchasing dataset can be an important new benchmark and our model shades light for future works on round-based AI reasoning.

2 Related Work

Learning to Learn & Few-Shot Learning Existing learning to learn or meta-learning studies (Hochreiter, Younger, and Conwell 2001; Thrun and Pratt 2012) mainly focus on supervised learning problems. A particularly challenging problem is learning with few training examples, i.e. few-shot learning. Generally, few-shot learning datasets contain several tasks, and for each task, there are a limited number of examples with supervised information. Few-shot learning algorithms can improve on new tasks using provided supervision. Within the learning process, the meta knowledge is extracted by a meta-learner, which learns to generalize the meta knowledge on each specific task. Vinyals et al. (2016) uses Matching Networks with attention and memory to enable rapid learning. Snell, Swersky, and Zemel (2017) propose Prototypical Networks. While Ravi and Larochelle (2016) use an LSTM-based meta-learning to learn an update rule for training a neural network learner, Model-Agnostic Meta-Learning (MAML) (Finn, Abbeel, and Levine 2017) learns a good model parameter initialization that can quickly adapt to similar tasks. Reptile (Nichol and Schulman 2018) is a first-order approximation of MAML, which is remarkably simple and performs similarly well. In this study, we adapt Reptile to our framework.

Gaming Machine Learning Datasets Datasets and environments are crucial for facilitating gaming machine learning research by serving as benchmarking platforms for new methods. STARDATA (Lin et al. 2017) is currently the largest StarCraft AI Research Dataset, dedicated to real-time strategy (RTS) games research. Hu et al. (2019) provides a simpler RTS environment that focus more on language instruction as macro-actions. For first-person shooter (FPS) games, VizDoom (Kempka et al. 2016) provides an environment for 3D visual Reinforcement Learning. MineRL (Guss et al. 2019) introduces a large-scale, simulator-paired dataset of human demonstrations of sandbox game MineCraft. The Atari games are popular for RL methods evaluation, and The Atari Grand Challenge Dataset (Kurin et al. 2017) has catalyzed the research. These studies focus on continuous environments without interruptions. In our work, we introduce a novel round-based gaming dataset based on CS:GO. Each round can be considered as an independent gaming episode that equally contributes to the match where a multi-round meta-strategies exist. Players can strategically lose some gaming episodes in exchange for winning a long-term goal.

3 Task

3.1 Few-shot Learning

Each team could develop several multi-round economy strategies that deliberately let go of some disadvantaged rounds temporarily to save money and build up comparative advantages for future rounds. As for each game, each player has its own preference of weapon purchasing. The financial status profoundly impacts the player's purchasing policy in each round. Due to the complexity and diversity of each player's attributes (policies), we cast the task into the learning-to-learn framework. For each game, the first few rounds are observed, and the model learns to predict later rounds. This few-shot task setting brings more opportunities for agents to learn players' dynamic attributes during inference and challenge the agents to learn more generalized policies that can quickly be adapted to current players after some observations.

3.2 Problem Formulation

We treat each match as a separate data point. Each match consists 10 different tasks from 10 players' perspectives. Since each player has its own preference for weapons, we formulate the problem into a few-shot learning task to foster the agent to capture the preference from the few support shots. For each match M_i, each player i go through j rounds, rounds are noted as $R_{i,j}, j \in [1, n_i]$, $M_i = \{R_{i,1}, R_{i,2}, ..., R_{i,n_i}\}$. We use the first K rounds as K-shot training examples. The model adjusts on the K shots (support set) and is asked to behave well on other $n_i - K$ rounds (target set).

We formulate the problem in a reinforcement learning setup. At the beginning of each round, an agent (a player) estimates the states and takes a single action that stands for a weapon purchasing set. The state of the agent includes weapons and money of all players.

We introduce the formulation for a match M_i and for simplicity, we omit the subscript i. For the match M, the agent's possessions at the j-th round is composed of the weapons it owns $X_j = \{x_{j,1}, x_{j,2}, \ldots, x_{j,m_j}\}$ and money c_j. At round j, the history information of this round $H_j = \{E_1, E_2, \ldots, E_{j-1}\}$ contains empirical information of past rounds. The empirical information of j-th round E_j consists of final weapons after purchasing $X'_j = \{x'_{j,1}, x'_{j,2}, \ldots\}$ and performance score s_j . For the j-th round, given an agent's own weapons X^s_j, team's weapons $X^t_j = \{X^t_{j,1}, X^t_{j,2}, X^t_{j,3}, X^t_{j,4}, X^t_{j,5}\}$ and opponent's weapons $X^o_j = \{X^o_{j,1}, X^o_{j,2}, X^o_{j,3}, X^o_{j,4}, X^o_{j,5}\}$, along with history information $H_j = \{E_1, E_2, \ldots, E_{j-1}\}$ from past rounds and all players' money, the agent needs to properly generate the action A_j to approach the label $\hat{A}_j$.

4 Dataset

We aim to build a dataset dedicated for round-based game reasoning. To build our dataset, we collect professional CS:GO players' match-replays during 2019.

4.1 Parsing Replays

We design a systematic procedure to process the replays. First, we parse the replays using the demofile parser[2]. We then filter out anomalous data to ensure quality. For all the

[2]https://saul.github.io/demofile/index.html

Attribute	Description
account	current cash
cash spent	cash the player has spent this round
weapons	all weapons held by this player
items value	sum of current items' prices
performance score	player's score in the scoreboard

Table 1: Description of the extracted information of a player for each round. Performance score is described in Section 4.1.

Type \ Count	0	1	2	3	4
Gun	35.9%	61.6%	2.4%	0.1%	0%
Grenade	19.4%	12.6%	14.6%	16.4%	37.0%
Equipment	38.3%	50.3%	10.7%	0.7%	0%

Table 2: The distribution of purchasing action count for each type of weapon. Note that gun and equipment purchased 4 times in one purchasing sequence are very rare so they are rounded down and visualized as 0% in this table.

replays, we extract all information related to weapon purchasing. Specifically, we capture players' weapon pickup and weapon removal actions. We also extract each player's state 3 times each round, including round start, weapon purchasing period end, and round end. Table 1 shows detailed description for the extracted states. The performance score is provided by the CS:GO in-game scoreboard, which is based on the player's kill and bomb planted/defused. We use the normalized score in our Round Attribute Encoder for past rounds' encoding.

After parsing, we convert them into structured JSON format. Data cleaning is subsequently performed to ensure data quality. We drop out the data with inconsistencies on weapon purchased and money spent. We then obtain consistent data of 5167 matches. We randomly shuffle the matches and split the data into training, development, and test set in the ratio 8:1:1. The training, development, and test set consist of 4133, 517, and 517 matches, respectively.

4.2 Statistics

In CS:GO, there are 44 different weapons in total, including 34 guns, 6 grenades and 4 equipments. For guns, there are 6 different types: pistols, shotguns, SMGs, Automatic Rifles, LMGs and Sniper Rifles. Equipment includes helmet, vest, defuse kit and Zeus x27. In order to get high-level representations for their intrinsically diverse attributes, we use a self-supervised learning method to train the embedding of weapons. We treat guns, grenades and equipment as three types and perform generation separately in our model.

Table 2 shows the frequency distribution of three categories of purchasable items, i.e., gun, grenade, and equipment, in the purchasing sequences per round. Table 2 does not contain information about the the exact position within a sequence. We sort all the weapon sequences in the order of gun, grenade, and equipment based on our human prior knowledge.

Algorithm 1 Our modified model agnostic reptile meta-learning algorithm

k = number of shots in few-shot learning.
ϵ = meta-learning step size.
Initialize model parameters θ.
repeat
  Sample single match data M_i with repetition
  for j = 1 **to** k **do**
    Sample single round R_{ij} and compute loss L
    $\theta' \leftarrow \theta - \nabla_\theta L$
  end for
  Compute target set loss L' on the other rounds.
  $\theta'' \leftarrow \theta' - \nabla_{\theta'} L'$
  Update $\theta \leftarrow \theta + \epsilon(\theta'' - \theta)$
until convergence

Although a player can only carry 1 primary gun and 1 pistol, there are some cases that players buy more than 2 guns, probably for their teammates. As in this paper, we only consider studying the task as a single agent round-based problem, we do not remove these cases and leave learning collaborative purchasing to future work. The player can carry 4 grenades and 1 of each type of equipment at maximum so the maximum action length for purchasing is 4. Each row shows for each type of weapon when in the purchasing sequence it is likely to be purchased.

5 Methodology

5.1 Meta-Learning Algorithm

The detailed algorithm is described in Algorithm 1. We have investigated MAML and its first-order simplification (Finn, Abbeel, and Levine 2017) and later moved to Reptile (Nichol and Schulman 2018) since it is also first-order and performs reasonably well in many tasks. Note that the original algorithm is used for image classification and each task/class contains several data. In our case, however, we have more tasks/games and fewer data to support. Thus, for target sets of training data, we update the model parameters as well while the original approach performs evaluations only after several epochs. For the meta-learning loop, we use vanilla SGD that samples a single task for each step. In the inner loop for adapting to each task, we use Adam (Kingma and Ba 2014) as optimizer.

5.2 Atomic Action and Embedding

Although each round only requires a one-time purchase to receive the result, the purchasing action space is huge and complex. It contains different combinations of 44 weapons for each player. We thus approximate the complex joint action by splitting it into a sequence of ***atomic actions*** and each atomic action represents a single weapon purchase. According authors' human priors, professional players mostly show the same pattern of purchasing weapons: they prefer to buy guns first, then grenades, and equipment at last if there are still money left. We therefore sort the action sequence by the specific order in accordance with the professional purchasing habit. Then we formulate the one-time purchasing

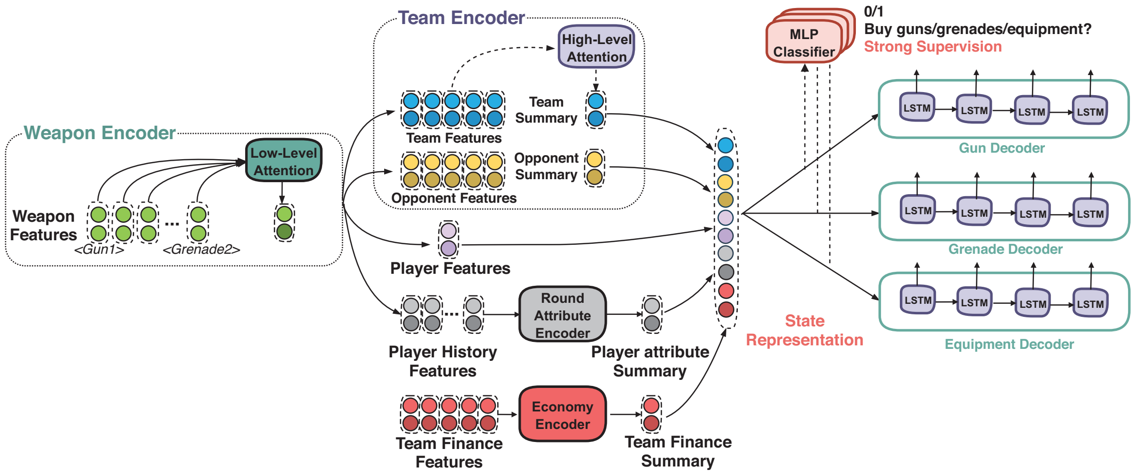

Figure 1: Model Architecture. Weapon encoder generates a weapon summary based on a set of weapon features of a player. Team encoder encodes a set of weapon summaries into a single team summary. Round attribute encoder encodes the player's information of previous rounds. The state representation is then fed into three LSTMs separately. Three gate classifiers are trained with strong supervision to determine if the generation of a certain type of action is suitable.

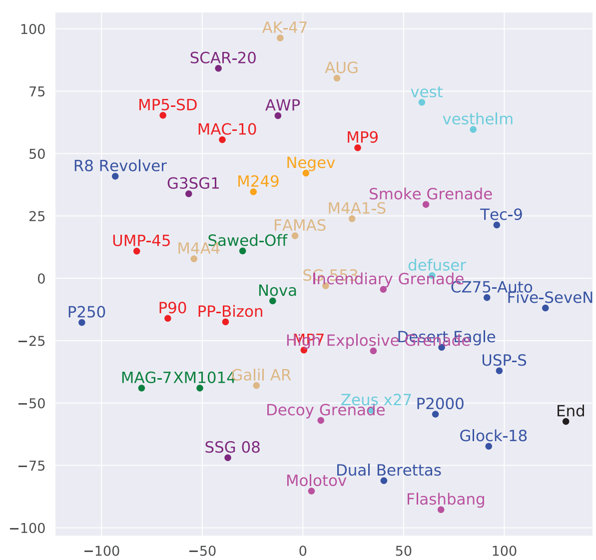

Figure 2: Atomic action Embedding t-SNE visualization. Different color stands for purchase different types of weapons. We further categorize guns into 6 types just for visualization.

action into a sequence of atomic actions and consider it as a sequence generation problem.

Therefore we can narrow down the action space. We pre-train the atomic action embeddings of purchasing each weapon by continuous bag-of-words model (Mikolov et al. 2013) using the context information in the training set. By visually inspecting the t-SNE of atomic action embeddings (Figure 2), we verify that purchasing similar weapons are more close to each other which can support the sequence decoding process. We highlight guns in different colors based on its subtype. The "End" action will terminate the purchase.

5.3 State Encoder

The state of the agent includes the weapons of itself, teammates and enemies. We use hierarchical attention (Yang et al. 2016) to represent the aggregated representation of each player and each team. By using hierarchical attention, we fuse the order-independent weapons representations into player representations and player representations into team representations. With the round attribute encoder, we leverage the round information by incorporating the weapon history and performance score of the agent at past rounds to force the agent to learn from the past rounds.

Weapon Encoder Not all weapons contribute equally to the representation of the player's attribute. Hence, we apply the attention mechanism to attend to more important weapons for the player and aggregate the representation of the weapons to vectorize a player. Specifically, weapon representation p_i for player i is

$$u_{i,t} = tanh(W_1 x_{i,t} + b_1), \tag{1}$$

$$\alpha_{i,t} = \frac{exp(v_1^T u_{i,t})}{\sum_t exp(v_1^T u_{i,t})}, \tag{2}$$

$$p_i = \sum_t \alpha_{i,t} x_{i,t}. \tag{3}$$

That is, we get the hidden representation $u_{i,t}$ from weapon embedding $x_{i,t}$ of t-th weapon of player i. We then measure the weight of the weapon $\alpha_{i,t}$ based on the similarity between $u_{i,t}$ and a weapon importance vector v_1^T. After that, the aggregated representation of weapons of one player is the weighted sum of the weapon embeddings. W_1, b_1 and v_1^T are trainable parameters, and shared for all players.

Team Encoder Similarly, we use the attention mechanism to assign weights to players in each team and thereby compute the aggregated team representation z.

Round Attribute Encoder In the proposed round-based task, previous rounds' actions and feedbacks are explicitly perceived and should be carefully reflected by the agent. Therefore, round information is a key feature that needs to be effectively utilized. Specifically, in the j-th round, the agent knows its final weapons after purchasing from previous rounds $\{X'_1, \ldots, X'_{j-1}\}$ and performance scores $\{s_1, ..., s_{j-1}\}$. The past weapons are passed into weapon encoder to get the aggregated weapon representation of previous rounds. Then we normalize the performance scores as weights and compute the weighted sum of past weapon representation as round representation h_r. As a result, the agent will attend to rounds in which it has better performance.

Economy Encoder To take the economy into account, we encode the normalized money features of all players through a multi-layer perceptron (MLP) as a dense economy vector h_c.

State Representation We concatenate the agent's player representation p^s, two teams' representation z^a, z^e, round representation h_r and money representation h_c. Then we get the overall initial state representation using MLP.

$$h = W_{h2}\ ReLU(W_{h1}[p^s; z^a; z^e; h_r; h_c]). \tag{4}$$

5.4 Multi-Task Decoder

Given the player's states, the agent is asked to take sequence of atomic actions and receive a reward from the environment. Such atomic actions can be classified into three categories by their types, i.e. purchasing guns, grenades or equipment. Since attribute of each type of weapon and the prices of weapons are diverse, the strategies for generating different types of weapons should be different. We formulate the purchasing as a multi-task atomic action sequence generation problem.

Gate Network Before the decoding step, we train three gate networks to control the atomic action generation of each task. Each gate is a binary classifier, a simple MLP, which decides whether to generate actions of a task. They are trained independently throughout the entire training procedure with strong supervision signals: whether a label has actions of this task. The gates can facilitate action generation and are easy to train.

Task-Specific Decoder Based on the initial state representation, the agent generates atomic actions sequentially and transits state using LSTM (Hochreiter and Schmidhuber 1997). Each LSTM-based decoder is designed for one task and all decoders share the same player money information and the initial state representation given by the encoder. This multi-task design also ensures a better generalization of the encoder with training signals of different domains (Caruana 1997).

The agent takes the state representation h to initialize the LSTM hidden state h_0 and use the hidden state h_t at each time step t to generate the distribution over atomic actions.

$$h_t = LSTM(h_{t-1}, a_{t-1}), \tag{5}$$

$$P(\cdot|a_1, ..., a_{t-1}, h) = \sigma(W_{c2}\ ReLU(W_{c1} h_t)). \tag{6}$$

where σ is the softmax function.

5.5 Learning Objective

Since we use LSTM to generate the atomic actions sequentially, using the "teacher forcing" algorithm (Williams and Zipser 1989) to train our model will inevitably result in the exposure bias problem (Ranzato et al. 2015): maximizing the likelihood of a sequence of atomic actions needs the ground truth atomic action sequence during training but such supervision signal is not available in testing, thus errors are accumulated while generating the atomic action sequence. To address the issue, we use the self-critical sequence training (SCST) method (Rennie et al. 2017). SCST is a form of REINFORCE (Williams 1992) algorithm that is designed for tackling sequence generation as RL problem. We first sample an atomic action $a^s_t \sim P(\cdot|a^s_1, ..., a^s_{t-1}, h)$ from the atomic action distribution at each generating step t to get an atomic action sequence A^s. Another sequence A^g is generated using greedy search to maximize the output probability distribution $P(\cdot|a^g_1, ..., a^g_{t-1}, h)$ at each step and serves as a baseline. We define $r(A)$ as the reward function. We compute $F1$ score with ground-truth as our reward function since we do not consider the atomic action sequence order and compare the formed atomic action set only. The objective function is defined as follows.

$$L = (r(A^g) - r(A^s)) \sum_t log\ p(a^s_t|a^s_1, ..., a^s_{t-1}). \tag{7}$$

Our gate network is trained separately with strong supervision signal. We compute the cross entropy loss for each binary gate.

5.6 Evaluation Metrics

We evaluate the methods by calculating the $\mathbf{F_1}$ **score** between the model output atomic action sequence and ground truth. Same as the reward function used in learning objective.

6 Experiments

In this section, we evaluate the proposed methods on the test set based on their highest performance on the development set. During atomic action generation step, we mask out weapons that cannot currently afford all methods to avoid invalid purchases in both training and testing phase. In addition, some weapons such as grenades has a quantity limit. We mask out weapons that currently reach the quantity limit as well.

We do an ablation study on our multi-sequence reasoner to measure its effectiveness.

Method	F_1	$Precision$	$Recall$	F_1-gun	F_1-grenade	F_1-equip
Greedy Algorithm	0.2612			0.0338	0.3831	0.2890
Single-Sequence Reasoner w/ Gate	0.5109			0.3487	0.5734	0.7119
Single-Sequence Reasoner + RAE w/ Gate	0.5206	0.5275	0.5332	0.3494	0.5870	**0.7177**
Multi-Sequence Reasoner w/o Gate	0.5028			0.3216	0.5912	0.5370
Multi-Sequence Reasoner + RAE w/o Gate	0.5114			0.3216	0.6008	0.5435
Multi-Sequence Reasoner w/ Gate	0.5475			**0.4524**	0.6608	0.5386
Multi-Sequence Reasoner + RAE w/ Gate	**0.5670**	0.5502	0.6161	0.3920	**0.6639**	0.6006

Table 3: The results of different methods including ablation study. We categorize weapon outputs by gun, grenade and equipment (equip) to get more insights. The multi-sequence reasoner with round attribute encoder (RAE) and gate classifier achieves the best result. Both gate classifier and RAE can improve model performance in different circumstances.

6.1 Greedy Algorithm Baseline

The greedy method buys weapons based on type order. For each type, it will do a sequence of purchases prioritized by weapon expensiveness. In other words, at each purchasing step, it buys the most expensive weapon which is affordable in that type. It only buys one gun, a maximum number of grenades up to the quantity limit, and then buy equipment. We consider it as the baseline as it does not require training and it is hard to generalize to different gaming scenarios.

6.2 Multi-Sequence Reasoner

The Multi-Sequence Reasoner follows the model architecture described in section 5. In the first round, all players start with no weapons and are restricted to buy pistols due to finance issues. Data in the first round are not generalizable and we cannot utilize its information for the second round. Since we need useful empirical information of past rounds and the second round only contains data from the first round, these two rounds are removed. In the 16th round, the two teams switch sides and start from scratch. Therefore 16th and 17th rounds are not included in our task as well.

We set the few shot number K to 5. We set the batch size to 10 to generate the atomic actions of ten players independently in a match at the same time. Note that we tackle this task as a single agent problem and leave team-based multi-agent purchasing prediction to future work. During inference time, we use the beam search to generate optimal atomic actions and set the beam size to 1.

Round Attribute Encoder We evaluate the effectiveness of round attribute encoder by concatenating its output to the state representation encoded by the original sequence reasoner without modifying the original model architecture. To measure its effectiveness, for all experiments, we run the model with two settings: with and without the round attribute encoder.

6.3 Result

We report the performance of different methods in Table 3. We also showed the performance of each type of weapon purchasing. First of all, we observe that the naive Greedy Algorithm does not achieve a good performance comparing to deep learning models. Besides, we also observe that multi-sequence reasoner with round attribute encoder in the last row of table 3 achieves the highest F_1.

Ablation Study To test the importance of the gate network (Gate), round attribute encoder (RAE), and multi-task decoder (Multi-Sequence Reasoner), we perform an ablation study where we remove the gate network, round attribute encoder, and turn multi-task decoder into single decoder (Single-Sequence Reasoner). As shown in Table 3, ablating gate network, round attribute encoder, and multi-task decoder from our integrated model will impair the performance and lead to a decrease of 5.56%, 1.95% and 4.64% for F_1 score. More importantly, consistently decreased performance in all three ablation models due to RAE removal shows the importance of utilizing round meta information. Thus, we believe round-based games are fundamentally different from conventionally studied continuous games. How to learn an effective round meta information representation and how to utilize it is an important topic for future game AI studies.

7 Conclusion

We explored the challenges in round-based games in which each data contains a long sequence of dependent episodes. We introduced a new round-based dataset based on CS:GO. The dynamic environment and connections between rounds make it suitable for round-based game study. We presented a few-shot learning task to encourage the agent to learn general policies and can quickly adapt to players' personal preferences in certain scenarios. Experimentally, we showed that our proposed model, Multi-Sequence Reasoner, is effective. We found that using round empirical information leads to nontrivial improvement on the result, thus testifying the importance of round history for the task. We believe our research will open doors for building interpretable AI for understanding episodic and long-term behavioral strategies not only for the gaming community but also for the broader online platforms.

8 Acknowledgements

The authors would like to thank Yang Yang, Sijie Duan and Qi Luo for their help on dataset collection. YZ and EF are grateful to DARPA (grant no. D16AP00115).

References

Caruana, R. 1997. Multitask learning. *Machine learning* 28(1):41–75.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1126–1135. JMLR. org.

Guss, W. H.; Houghton, B.; Topin, N.; Wang, P.; Codel, C.; Veloso, M.; and Salakhutdinov, R. 2019. Minerl: a large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hochreiter, S.; Younger, A. S.; and Conwell, P. R. 2001. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, 87–94. Springer.

Hu, H.; Yarats, D.; Gong, Q.; Tian, Y.; and Lewis, M. 2019. Hierarchical decision making by generating and following natural language instructions. In *Advances in neural information processing systems*, 10025–10034.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

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

Kurin, V.; Nowozin, S.; Hofmann, K.; Beyer, L.; and Leibe, B. 2017. The atari grand challenge dataset. *arXiv preprint arXiv:1705.10998*.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space.

Nichol, A., and Schulman, J. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2:2.

OpenAI; Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; Józefowicz, R.; Gray, S.; Olsson, C.; Pachocki, J.; Petrov, M.; de Oliveira Pinto, H. P.; Raiman, J.; Salimans, T.; Schlatter, J.; Schneider, J.; Sidor, S.; Sutskever, I.; Tang, J.; Wolski, F.; and Zhang, S. 2019. Dota 2 with large scale deep reinforcement learning.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Ravi, S., and Larochelle, H. 2016. Optimization as a model for few-shot learning.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7008–7024.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, 4077–4087.

Thrun, S., and Pratt, L. 2012. *Learning to learn*. Springer Science & Business Media.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, 3630–3638.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 1480–1489.