# Report for Practical Work in AI
# Multi-Agent Reinforcement Learning is A Sequence Modeling Problem

**Jenish Thapa**
Institute of Machine Learning
Johannes Kepler Universität Linz
k12137169@students.jku.at

## Abstract

This report delves into the practical implementation and reproduction of some selected benchmark results of the paper "Multi-Agent Reinforcement Learning is A Sequence Modeling Problem." [15]. This paper introduces an architecture named Multi-Agent Transformer (MAT) that effectively casts cooperative multi-agent reinforcement learning (MARL) into sequence modelling problem. Employing an encoder-decoder architecture, MAT capitalizes on the multi-agent advantage decomposition theorem. In this report, MAT was implemented, trained and tested along some significant MARL benchmarks. Additional to this, another recent sequential model "RWKV: Reinventing RNNs for the Transformer Era" [1] was adapted for MARL inspired from MAT and a novel architecture MARWKV (Multi Agent RWKV) was implemented, trained and evaluated on same benchmarks. This report sheds comparison on the implementation of these two architectures in context of multi-agent reinforcement learning. The project's repository can be found at https://github.com/j-thapa/MAT-MARWKV.git.

## 1 Introduction

Multi-Agent Reinforcement Learning (MARL) presents significant challenges, primarily due to the complexity of identifying and combining individual agents policy improvements in a way that benefits the entire team [16, 5]. Traditional MARL problems have been partly addressed by the approach of centralized training for decentralized execution (CTDE) [6, 17], which provides agents access to global information and opponents actions during training. However, these methods often fall short in capturing the intricate dynamics of multi-agent interactions, with some failing even in basic cooperative tasks [7]. To address these shortcomings, the multi-agent advantage decomposition theorem was proposed [7], offering a nuanced understanding of how different agents contribute to overall returns. This theorem lays the groundwork for cooperation through a sequential decision-making process.

Recently, sequence models (SMs) have made significant strides in the field of natural language processing (NLP) [10]. While primarily utilized in language tasks, the applicability of sequential methods extends beyond NLP, as a widely applicable general foundation model [2]. The architecture like Multi-Agent Transformer (MAT) bridges the gap between cooperative MARL challenges and sequence modeling. At the core of these architectures is the multi-agent advantage decomposition theorem and a sequential update scheme, which together transform the joint policy optimization of multiple agents into a sequential policy search process. In this report two recent relevant sequence model architectures transformer and RWKV [1] are applied in the context of MARL.

## 2 Related Works

### 2.1 Multi-Agent Advantage Decomposition theorem

The multi-agent advantage decomposition theorem [8] addresses the challenge of individual agents receiving a shared reward and struggling to discern their specific contribution to the team's success or failure [2]

Let $i_{1:n}$ be a permutation of agents. Then, for any joint observation $\boldsymbol{o} \in \boldsymbol{\mathcal{O}}$ and joint action $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, the following equation always holds with no further assumption needed,

$$A_{\boldsymbol{\pi}}^{i_{1:n}}(\boldsymbol{o}, \boldsymbol{a^{1:n}}) = \sum_{m=1}^{n} A_{\boldsymbol{\pi}}^{i_m}(\boldsymbol{o}, \boldsymbol{a^{i_{1:m-1}}}, a^{i_m}).$$

Suppose that agent $i_1$ picks an action $a^{i_1}$ with positive advantage, $A_{\boldsymbol{\pi}}^{i_1}(o, a^{i_1}) > 0$. Then, imagine that for all $j = 2, \ldots, n$, agent $i_j$ knows the joint action $a^{1:j-1}$ of its predecessors. In this case, it can choose an action $a^{i_j}$ for which the advantage $A_{\boldsymbol{\pi}}^{i_j}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j})$ is positive. Altogether, the theorem assures that the joint action $a^{1:n}$ has positive advantage. Furthermore, the joint action has been chosen in $n$ steps, each of which searched an individual agent's action space. Hence, the complexity of this search is additive in the sizes of the action spaces instead of direct joint space multiplicative size.

### 2.2 The Transformer Model

The Transformer architecture [14], maintains an encoder-decoder structure. The encoder processes the input sequence of tokens and converts them into latent representations. Subsequently, the decoder generates the desired output sequence in an auto-regressive manner. During each inference step, the Transformer utilizes all previously generated tokens as input. A crucial element of the Transformer is the scaled dot-product attention mechanism, which effectively captures the interrelationships within input sequences.

### 2.3 The RWKV model

The RWKV model, short for Receptance Weighted Key Value, is a recurrent neural network (RNN) architecture. It combines the best of both RNN and transformer architecture. RWKV supports decoding like other RNN networks for longer sequences i.e unlike transformer it doesn't need to address all previously generated tokens as input, the hidden state stores the past information. While training it can be train partially in parallel which is achieved by splitting the full RNN network into multiple smaller layers, where each layer's hidden state can be used independently to compute the next token hidden state for the same layer.

## 3 Method

Python is the primary programming language for the project. The PyTorch framework is used for training, loading, and evaluating models. The MAT algorithm, along with environment wrappers, data collectors, data loaders, and utilities for training and evaluation, were implemented following the official MAT repository [9]. Benchmark environments were set up according to the instructions provided in both their own and the MAT repository documentation. The implementation code for RWKV, adapted for MARWKV, was sourced from the RWKV paper's repository [11], which also includes CUDA code for parallel computation of sequences. For logging training and evaluation data, TensorBoard was utilized, while pandas and matplotlib were used to generate line charts from the logged information.

### 3.1 Multi Agent Transformer (MAT)

MAT architecture suggested in the paper has been implemented, trained and evaluated. MAT has an encoder to learn representations from joint observations of agents and a decoder to sequentially generate actions for each agent in an autoregressive manner.

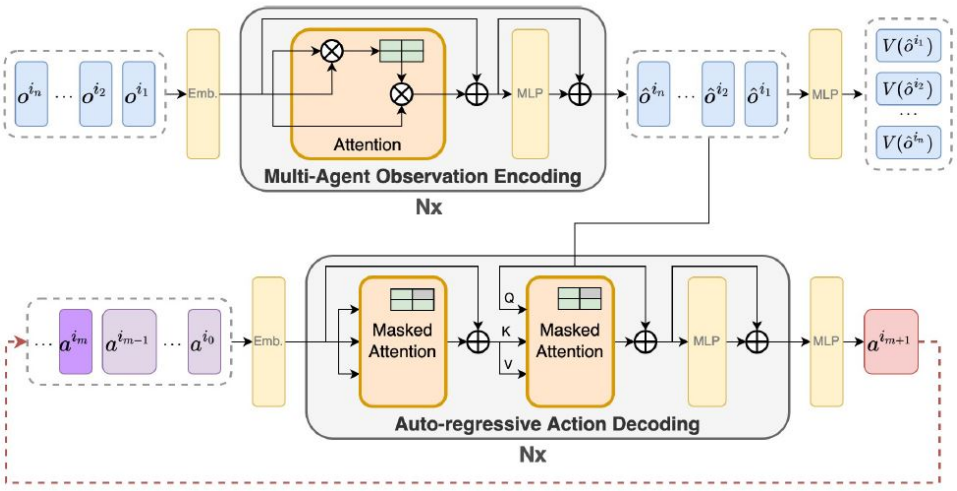


Figure 1: The encoder-decoder architecture of MAT [15].

**The encoder** processes a sequence of observations $(o^{i_1}, ..., o^{i_n})$ in an arbitrary order through multiple computational block. Each block includes a self-attention mechanism and a multi-layer perceptron (MLP), with residual connections. The resulting encoded observations, encapsulate both individual agent information and the high-level inter-agent relationships. For expressive representations, the encoder, during training, is tuned to approximate the value functions by minimizing the empirical Bellman error.

**The decoder** processes embedded joint action. Every decoding block has a masked self attention mechanism, where masking makes sure that for every $i_j$, attention is computed only between $i_r^{th}$ and $i_j^{th}$ action heads only when $r < j$, to maintain sequential update scheme. This is then followed by second masked attention, that computes attention between action heads and observation representations obtained from encoder. The block finished with a MLP and skipping connections. The last decoder block ouputs a sequence of representation of joint action which is feed to an MLP which outputs the probability distribution of $i_m$ action. Decoder is trained by minimizing clipping the Proximal Policy Optimization (PPO) [13] objective.

### 3.2 Multi Agent RWKV (MARWKV)

For analysis and comparison purposes, Multi Agent RWKV (MARWKV) has been developed as an alternative sequence model to the Transformer-based MAT. In this model, RWKV computational blocks replace the transformer blocks in both the encoder and decoder. Unlike the MAT, where observations and agent actions sequences are processed in parallel, MARWKV treats observations and actions sequences in a sequential manner, similar to traditional Recurrent Neural Networks (RNNs). The encoder sequentially processes observations from agents $(o^{i_1}, ..., o^{i_n})$ of arbitrarily order, producing encoded observations. These outputs from the encoder are then added with the encoded action input and feed to the decoder sequentially which are further processed through the RWKV computational blocks within the decoder which output the next agents action. Consistent with the approach in MAT, the encoder also approximate value function for observations and is fine-tuned by minimizing empirical Bellman error, while the decoder is trained by minimizing clipping PPO objective.

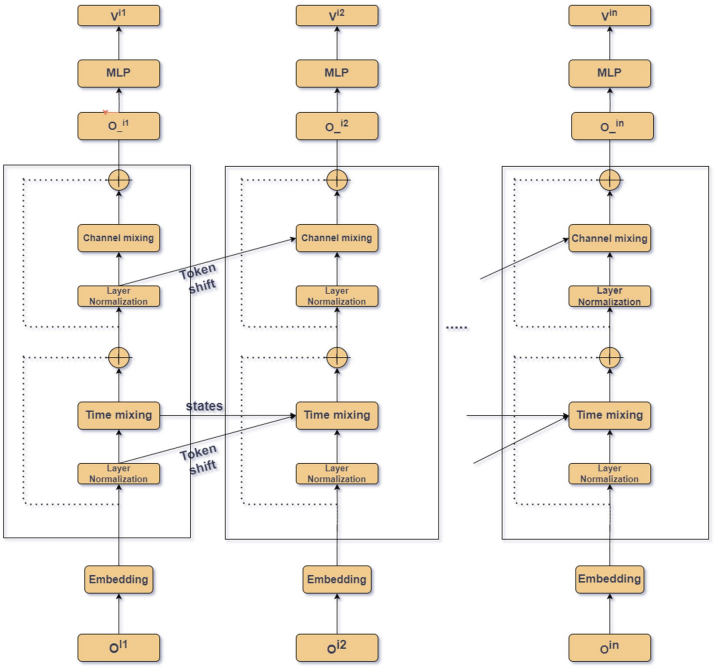


Figure 2: The encoder architecture of MARWKV.

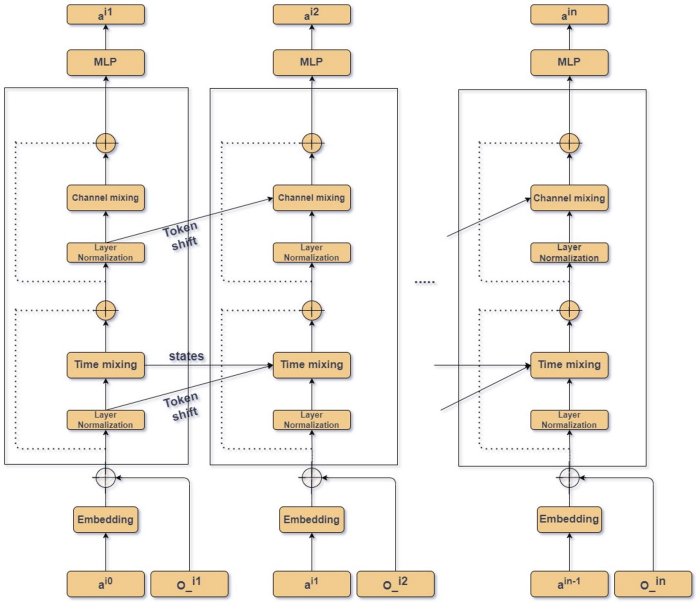


Figure 3: The decoder architecture of MARWKV.

RWKV computational block has two sub-blocks time-mixing and channel mixing. Within the time-mixing sub-block, the present input is interpolated with the previous time step input through the receptance vector R and positional weight decay vector W. This sub-block conducts the WKV computation, which is similar to the Transformer's attention, summing up weighted positions over the interval [1, t] using the activated receptance vector $\sigma$ (R), which aids in capturing the sequence's long-term dependencies. In the channel-mixing sub-block, elements R, K, and V are used, as in the time-mixing counterpart, where the current and previous inputs are interpolated linearly. The

resulting R, after being passed by a squared ReLU activation, is then multiplied with the weighted K to derive the output.

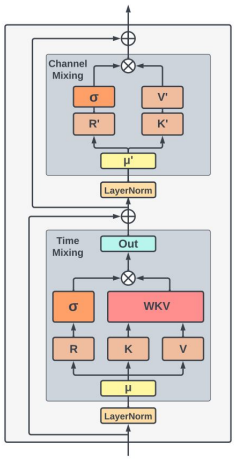


Figure 4: RWKV computational block.[1]

# 4 Experimental details and Results

## 4.1 StarCraftII Multi-Agent Challenge (SMAC)

SMAC is based on Blizzard's Starcraft II RTS game [12]. Here each unit in the game is controlled by an individual RL agent. The results of MAT on selected SMAC maps were reproduced. Along with MAT, MARWKV models were also trained on those maps. These maps varies in their setup, number of agents and hence difficulties.


Figure 5: Demonstration of SMAC environment tasks. [9]

For the study, four SMAC maps—two classified as hard and two as very hard—were selected to train and evaluate MAT and MARWKV algorithms. Analysis of the paper results reveals that MAT consistently outperforms MAPPO with parameter sharing [7], demonstrating superior efficacy in SMAC tasks. Both MAT and MAPPO shows a remarkable advantage over the HAPPO algorithm, where each agent operates under a distinct policy [18]. Our findings indicate that the MARWKV implementation achieves comparable performance to MAT across the varied SMAC maps selected. Given the homogeneous nature of SMAC agents, wherein they are interchangeable and capable of learning from the experiences of their teammates, we observe a significant increase in sample efficiency for MAPPO compared to HAPPO, leading to improved outcomes. Similarly, this sample efficiency is present in both MAT and MARWKV and they demonstrate substantial effectiveness in handling homogeneous tasks like those presented in SMAC.

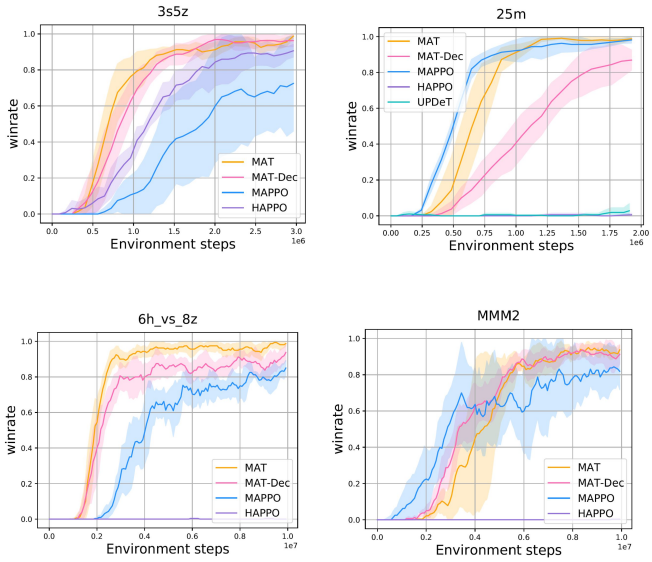


Figure 6: Performance comparisons of MAT and other algorithms on SMAC tasks. [15]

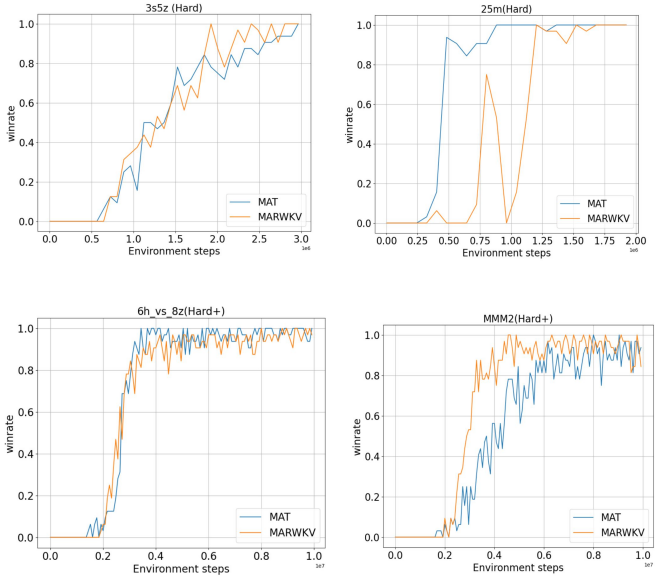


Figure 7: Performance comparisons of MAT and MARWKV on SMAC tasks.

### 4.2 Multi-Joint dynamics with Contact (MuJoCo)

MuJoCo is a general purpose physics engine that provides fast and accurate simulation of articulated structures interacting with their environment [4]. Halfcheetah is one of the scenario in the Mujoco environment where a simulation of half cheetah with six body parts or hinges as six different agents are done. In the study, the performance of algorithms was assessed not only on the standard half cheetah model but also under conditions where various joints were intentionally disabled. The methodology for disabling the joints, however, was not explicitly detailed in the paper. For this implementation, disabled joints were approached by disregarding the action input intended for the disabled joint and substituting it with a zero input.

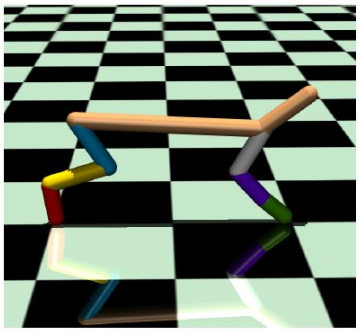

Figure 8: Demonstration of MuJoCo HalfCheetah. [15]

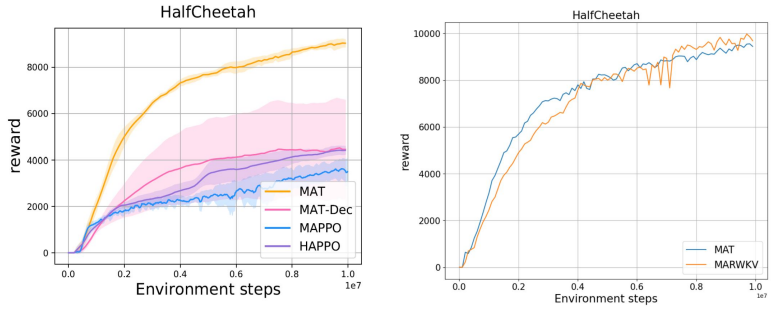


Figure 9: Performance comparison of MAT, MAPPO, HAPPO [15] and MARWKV on HalfCheetah.

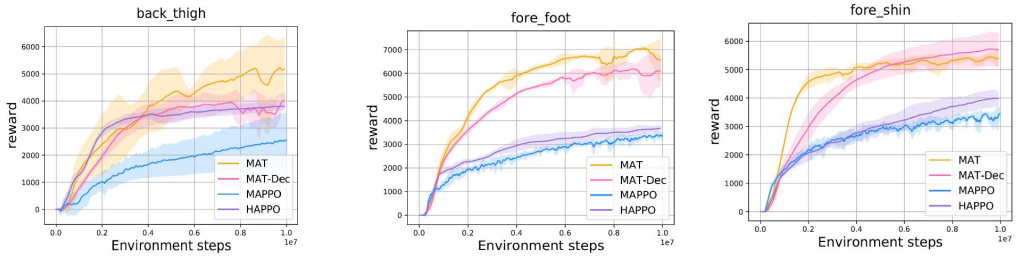


Figure 10: Performance comparison of MAT and other algorithms with different disabled joints HalfCheetah [15].

The results highlighted in the paper clearly demonstrate that MAT surpasses the performance of existing algorithms. Additionally, in the conducted experiments, MARWKV performed on par with MAT in simulations involving both disabled and fully enabled joints of the HalfCheetah model.

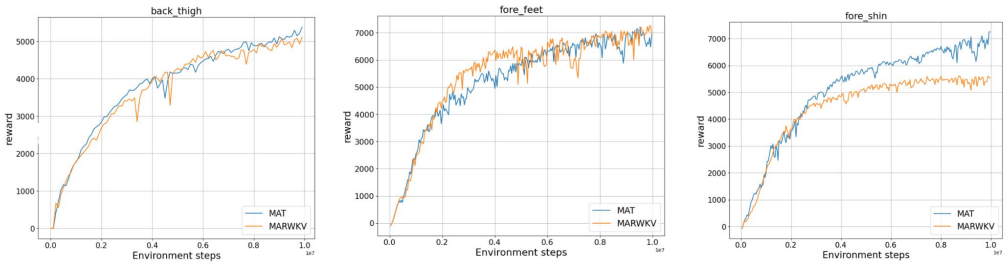


Figure 11: Performance comparison of MAT and MARWKV with different disabled joints HalfCheetah.

In these experiments, HAPPO exhibited superior performance compared to MAPPO. This outcome is attributed to the heterogeneous nature of the Mujoco environment, where training an agent responsible for foot movements with the experience of an agent controlling the thigh could be detrimental due to their distinct functions in the cheetah model. In such scenarios, HAPPO's approach of employing separate policies for each agent proved more effective than MAPPO's approach of sharing parameters. This leads to the conclusion that, similar to MAT, MARWKV also possesses significant modeling capabilities for handling heterogeneous tasks.

### 4.3 Bimanual Dexterous Manipulation (Bi-DexHands)

Bi-DexHands provides a collection of bimanual dexterous manipulations tasks and reinforcement learning algorithms [3] Bi-DexHands is built in the NVIDIA Isaac Gym. Like in MuJoCo agents in Bi-DexHands agents (i.e., joints, fingers, hands,...) are genuinely heterogeneous. Two environments, ShadowHandDoorOpenInward and ShadowHandDoorCloseOutward, were chosen for the training and evaluation of the MAT and MARWKV algorithms. These scenarios necessitate the use of both hands to open a door that is closed inward and to close a door that is open outward, respectively.

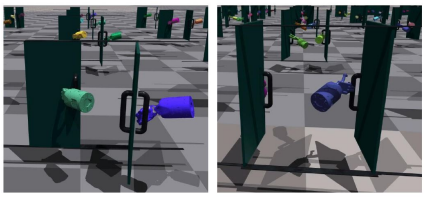


Figure 12: Demonstration of Bidex hands environment tasks. [15]

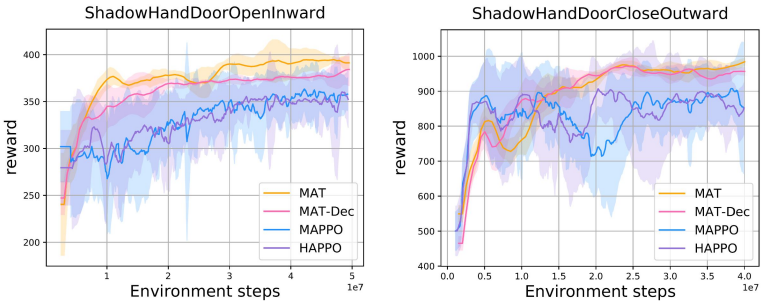


Figure 13: Performance comparisons of MAT and other algorithms on Bidex tasks.[15]

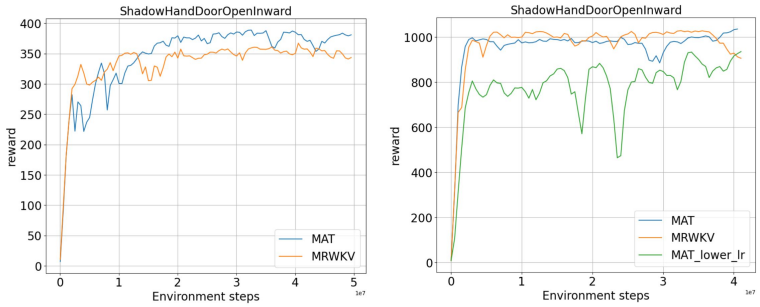


Figure 14: Performance comparisons of MAT and MARWKV on Bidex tasks.

The results show that MAT surpasses the performance of other existing algorithms in these bimanual hand tasks, and that MARWKV's performance matches that of MAT closely on both tasks. Notably, during the ShadowHandDoorCloseOutward task experiments, an increased learning rate of 5e-4 was applied to both MAT and MARWKV, diverging from the original learning rate of 5e-5 specified for MAT in the published paper. This adjustment was necessary because adhering to the lower learning rate from the paper did not yield the superior performance results for MAT that were documented in the original experiments.

### 4.4 Hyperparameter Settings for Experiments

The implementation and selection of hyperparameters for MAT are consistent with the official paper and its corresponding repository, with the sole exception being the learning rate used for the ShadowHandDoorCloseOutward task in the bimanual hands environment. Similarly, MARWKV shares the same hyperparameter configuration as MAT for nearly all experiments. However, an exception is made in the case of the SMAC learning rates, where MARWKV demonstrates improved performance with a higher learning rate. The hyperparameters chosen for the various tasks and algorithms are detailed in the tables provided below.

Table 1: Common hyperparameters used for MAT and MARWKV in the SMAC domain.

| **hyperparameters** | **value** | **hyperparameters** | **value** | **hyperparameters** | **value** |
|---|---|---|---|---|---|
| gain | 0.01 | optim eps | 1e-5 | batch size | 3200 |
| training threads | 16 | num mini-batch | 1 | rollout threads | 32 |
| entropy coef | 0.01 | max grad norm | 10 | episode length | 100 |
| optimizer | Adam | hidden layer dim | 64 | use huber loss | True |
| use gae | True | num blocks | 1 | num heads | 1 |
| stacked frames | 1 | | | | |

Table 2: Learning rate used for MAT and MARWKV in the SMAC domain.

| **Learning rate** | **algorithm** | **value** |
|---|---|---|
| actor | MAT | 5e-4 |
| critic | MAT | 5e-4 |
| actor | MARWKV | 1e-3 |
| critic | MARWKV | 1e-3 |

Table 3: Different hyperparameters used for MAT and MARWKV in the SMAC domain.

| maps | ppo epochs | ppo clip | steps | $\gamma$ |
|---|---|---|---|---|
| 3s5z | 10 | 0.05 | 3e6 | 0.99 |
| 25m | 15 | 0.05 | 2e6 | 0.99 |
| MMM2 | 5 | 0.05 | 1e7 | 0.99 |
| 6h vs 8z | 15 | 0.05 | 1e7 | 0.99 |

Table 4: Common hyperparameters used for MAT and MARWKV in the multi-agent MuJoCo domain.

| hyperparameters | value | hyperparameters | value | hyperparameters | value |
|---|---|---|---|---|---|
| gamma | 0.99 | steps | 1e7 | stacked frames | 1 |
| gain | 0.01 | optim eps | 1e-5 | batch size | 4000 |
| training threads | 16 | num mini-batch | 40 | rollout threads | 40 |
| entropy coef | 0.001 | max grad norm | 0.5 | episode length | 100 |
| optimizer | Adam | hidden layer dim | 64 | use huber loss | True |
| critic lr | 5e-5 | actor lr | 5e-5 | ppo epochs | 10 |
| ppo clip | 0.05 | num blocks | 1 | num head | 1 |

Table 5: Common hyperparameters used for all methods in the Bi-DexHands domain.

| hyperparameters | value | hyperparameters | value | hyperparameters | value |
|---|---|---|---|---|---|
| gamma | 0.96 | training threads | 16 | stacked frames | 1 |
| gain | 0.01 | optim eps | 1e-5 | ppo epochs | 5 |
| ppo clip | 0.2 | num mini-batch | 1 | rollout threads | 80 |
| batch size | 6000 | episode length | 75 | optimizer | Adam |
| entropy coef | 0.001 | max grad norm | 0.5 | num blocks | 1 |
| num heads | 1 | | | | |

Table 6: Different hyperparameters used in the Bi-DexHands domain.

| hyperparameters | ShadowHandDoorCloseOutward | ShadowHandDoorOpenInward |
|---|---|---|
| critic lr | 5e-4 | 5e-5 |
| actor lr | 5e-4 | 5e-5 |
| steps | 4e7 | 5e7 |

### 4.5 Performance Analysis of Different RWKV Versions

Three distinct versions 4, 5, and 6 of the RWKV computational block were implemented and evaluated within SMAC and MuJoCo benchmarks. To be more precise v4-neo is being used which is a bit more optimized version 4. It was observed that version 4 outperformed versions 5 and 6, in both environments, leading to its selection as the RWKV component in the MARWKV architecture. These variants were sourced and adapted from the RWKV's official GitHub repository [11], and identical hyperparameters were employed during the training phase for all three versions.

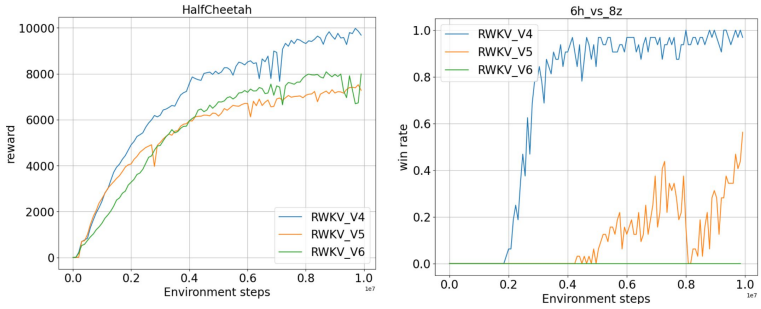


Figure 15: Performance evaluations of RWKV v4, v5 and v6 in SMAC and MUJOCO.

### 4.6 Training Speed and Performance

One significant advantage of RWKV is it's ability to be trained in parallel, making it highly efficient in terms of training speed, similar to the Transformer model. In the official implementation of RWKV, to facilitate parallel processing on GPUs, specialized GPU code written in CUDA is utilized. From the charts, it's evident that MARWKV is just slightly computationally slower compared to MAT while training in these two environments. Here the number of parameters in MARWKV is about 40,500 more than in MAT. Without the parallel CUDA code, the training speed of MARWKV is significantly reduced. Also, bfloat16 data type should be used in CUDA code to get efficient training speed.

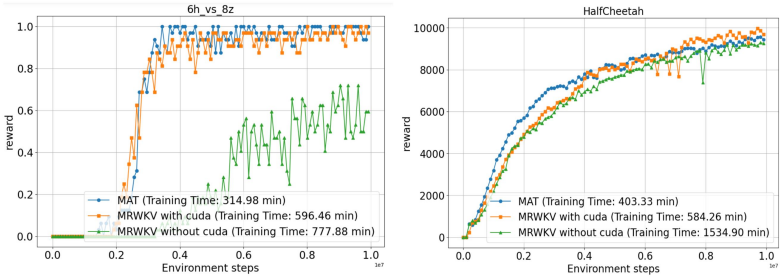


Figure 16: Training duration of MAT, MARWKV with CUDA code and MARWKV without CUDA code.

In the chart of SMAC (6h vs 8z) scenario, MARWKV performance is notably inferior to that of MAT and MARWKV with CUDA code, due to the use of a less aggressive learning rate. Conversely, employing a steeper learning rate in MARWKV within SMAC markedly enhances its performance.

### 4.7 Few Shot Learning Capabilities

To explore and compare the few-shot learning abilities of MARWKV and MAT, experiments were conducted using MuJoCo and SMAC, inspired from the experimental guidelines outlined in the paper. In the multi-agent MuJoCo setting, MAT and MARWKV models trained on the complete HalfCheetah robot was tested on three new tasks, each involving a different joint being disabled. To compare the pretrained models performance, MAT and MARWKV models were also trained from the scratch in those same tasks with equal amount of data as the control group to demonstrate the effectiveness of pre-training process.

Both MAT and MARWKV has very good generalization capabilities. The pretrained models outperfrom significantly compare to the models trained from the scratch.

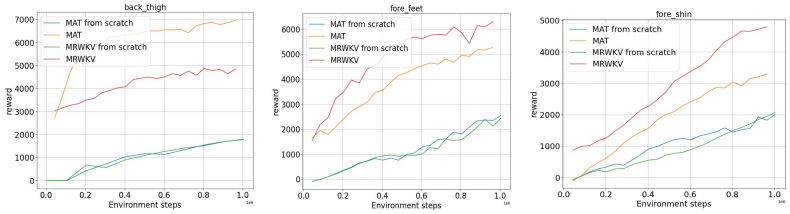


Figure 17: Few shot performance of MAT and MARWKV in MuJoCo

Initially, SMAC agents underwent training across eight different tasks featuring five distinct units, totaling 10 million steps within the environment. Subsequently, they were evaluated on four additional, more demanding tasks involving seven different units. The outcomes of this study reveal that while both MAT and MARWKV exhibit enhanced performance compared to models trained from scratch following pretraining, MAT notably surpasses MARWKV by a significant margin in this specific experiment.

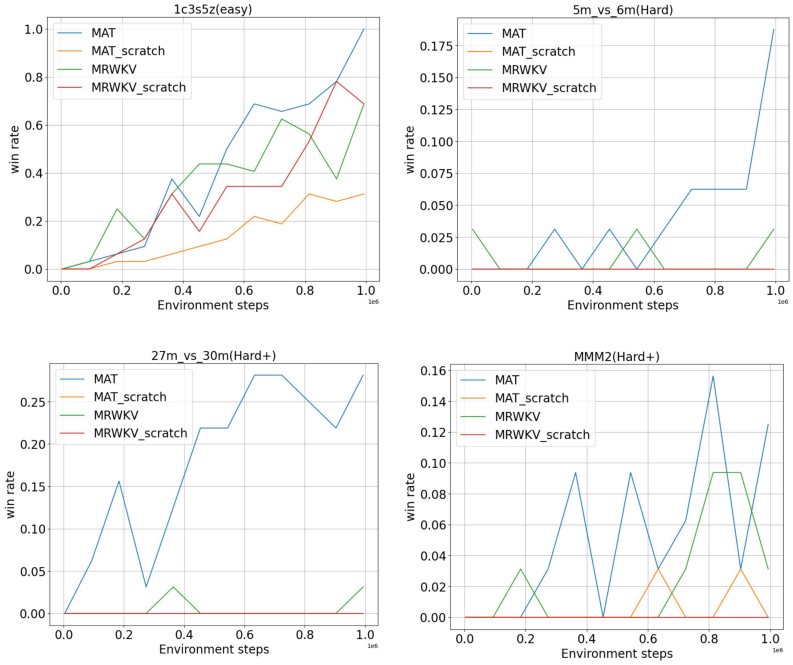


Figure 18: Few shot performance of MAT and MARWKV in SMAC

## 5 Conclusion

This report validates and examines the findings related to the Multi-Agent Transformer (MAT), designed to bridge the gap between multi-agent reinforcement learning (MARL) problems and sequence models (SM). Additionally, a relatively new efficient sequence model, RWKV, adapted for MARL as MARWKV was introduced. MARWKV demonstrates comparable performance to MAT, surpassing existing MARL algorithms in both homogeneous and heterogeneous environments. These results underscore the significant improvements in performance are achievable in MARL by applying sequence models, utilizing the multi-advantage decomposition theorem. Moreover, MARWKV not only matches MAT in performance and training parallelizability but also being a RNN model offers the added advantage of lower inference costs when generating agent actions sequentially. While the experiments performed in this report has comparatively very few number of agents, this advantage can be significant in scenarios involving a large number of agents. In such cases, incorporating all preceding agents action for attention in MAT becomes computationally expensive for generating

subsequent agents action. This highlights MARWKV suitability for large-scale MARL applications. Looking ahead, there's potential for integrating MAT and MARWKV into a novel architecture that harnesses the strengths of both models. Specifically, the robust observation encoding provided by MAT's attention mechanism could be combined with the computationally efficient RNN decoding of MARWKV. This could result in a more efficient system, capitalizing on the detailed observation analysis from MAT and the efficient sequential decoding process offered by MARWKV.

## References


[1] Peng Bo, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, and Xin Cheng et al. Rwkv: Reinventing RNNs for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, et al. On the opportunities and risks of foundation models. *arXiv preprint*, arXiv:2108.07258, 2021.

[3] Yuanpei Chen, Yaodong Yang, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuang Jiang, Stephen Marcus McAleer, Hao Dong, Zongqing Lu, and Song-Chun Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. *arXiv preprint arXiv:2206.08686*, 2022.

[4] Christian Schröder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *CoRR*, abs/2003.06709, 2020.

[5] Xiaotie Deng, Yuhao Li, David Henry Mguni, Jun Wang, and Yaodong Yang. On the complexity of computing markov perfect equilibrium in general-sum stochastic games. *arXiv preprint arXiv:2109.01795*, 2021.

[6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[7] Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *ICLR*, 2022.

[8] Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, and Yaodong Yang. Settling the variance of multi-agent policy gradients. In *Advances in Neural Information Processing Systems*, volume 34, pages 13458–13470, 2021.

[9] Wen Muning, Ji Jiaming, and Yang Y. Official implementation of MAT. `https://github.com/PKU-MARL/Multi-Agent-Transformer`, 2022.

[10] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[11] Bo Peng. Rwkv-lm: Repository for rwkv language model. `https://github.com/BlinkDL/RWKV-LM`, 2023.

[12] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge, 2019.

[13] John Schulman, F Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. `https://arxiv.org/abs/1707.06347`, 2017.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[15] M Wen, J Kuba, R Lin, W Zhang, Y Wen, J Wang, and Y Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2022.

[16] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[17] Yaodong Yang, Ying Wen, Jun Wang, Liheng Chen, Kun Shao, David Mguni, and Weinan Zhang. Multi-agent determinantal q-learning. In *Conference on Machine Learning*, pages 10757–10766. PMLR, 2020.

[18] Chao Yu, A. Velu, Eugene Vinitsky, Yu Wang, A. Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *ArXiv*, abs/2103.01955, 2021.