Appendices

In this appendix, we give the pseudocode of multi-agent planning and multi-agent trajectory prediction with MADIFF model in Section A. In Section B, we demonstrate how multiple agents' trajectories are modeled by MADIFF during centralized control and decentralized execution in an example three-agent environment. In Section C, we provide details of the experiments, including the normalization used to compute the average score, the detailed network illustration unrolling each agent's U-Net, crucial hyperparameters, and an example of wall-clock time required for training MADIFF. In Section D, we demonstrate and analyze additional experimental results. Specifically, we provide ablation results to support the effectiveness of opponent modeling in MADIFF-D, show the quality of opponent modeling by MADIFF-D on SMAC tasks, and visualize predicted multi-player trajectories by MADIFF and the baseline algorithm on the NBA dataset.

A ALGORITHM

Algorithm 1 Multi-Agent Planning with MADIFF

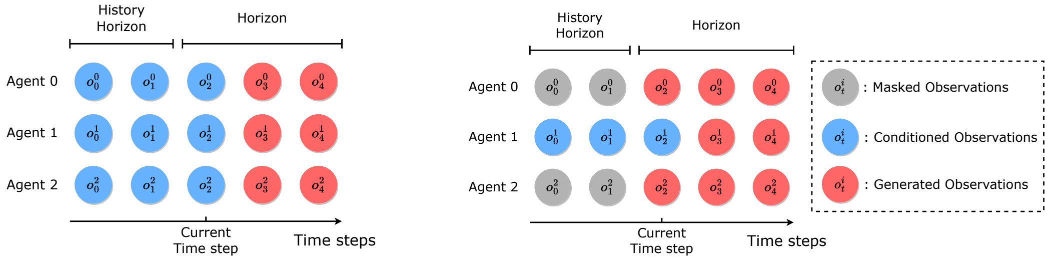
```
Input: Noise model $\epsilon_\theta$, inverse dynamics $I_\phi$, guidance scale $\omega$, history length $C$, condition $\boldsymbol{y}$
Initialize $h \leftarrow \texttt{Queue}(\texttt{length} = C)$;  $t \leftarrow 0$                // Maintain a history of length C
while not done do
    Observe joint observation $\boldsymbol{o}$;  $h.\texttt{insert}(\boldsymbol{o})$;  Initialize $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \alpha\boldsymbol{I})$
    for $k = K \ldots 1$ do
        $\boldsymbol{\tau}_k[:\texttt{length}(h)] \leftarrow h$                // Constrain plan to be consistent with history
        if Centralized control then
            $\hat{\boldsymbol{\epsilon}} \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k) + \omega(\epsilon_\theta(\boldsymbol{x}_k(\tau), \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{\tau}_k, k))$                // Classifier-free guidance
            $(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \hat{\boldsymbol{\epsilon}})$
        else if Decentralized execution then
            for agent $i \in \{1, 2, \ldots, N\}$ do
                $\hat{\epsilon}^i \leftarrow \epsilon^i_\theta(\tau^i_k, k) + \omega(\epsilon^i_\theta(\tau^i_k, y^i, k) - \epsilon^i_\theta(\tau^i_k, k))$                // Classifier-free guidance
                $(\mu^i_{k-1}, \Sigma^i_{k-1}) \leftarrow \texttt{Denoise}(\tau^i_k, \hat{\epsilon}^i)$
            end for
        end if
        $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_{k-1}, \alpha\boldsymbol{\Sigma}_{k-1})$
    end for
    Extract $(\boldsymbol{o}_t, \boldsymbol{o}_{t+1})$ from $\boldsymbol{\tau}_0$
    for agent $i \in \{1, 2, \ldots, N\}$ do
        $a^i_t \leftarrow f_{\phi^i}(o^i_t, o^i_{t+1})$
    end for
    Execute $\boldsymbol{a}_t$ in the environment;  $t \leftarrow t + 1$
end while
```

Algorithm 2 Multi-Agent Trajectory Prediction with MADIFF

```
Input: Noise model $\epsilon_\theta$, guidance scale $\omega$, condition $\boldsymbol{y}$, historical joint observations $h$ with length
    $C$, predict horizon $H$
Initialize $\boldsymbol{\tau}_K \sim \mathcal{N}(\mathbf{0}, \alpha\boldsymbol{I})$
for $k = K \ldots 1$ do
    $\boldsymbol{\tau}_k[: C] \leftarrow h$                // Constrain prediction to be consistent with history
    $\hat{\boldsymbol{\epsilon}} \leftarrow \epsilon_\theta(\boldsymbol{\tau}_k, k) + \omega(\epsilon_\theta(\boldsymbol{\tau}_k, \boldsymbol{y}, k) - \epsilon_\theta(\boldsymbol{\tau}_k, k))$                // Classifier-free guidance
    $(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1}) \leftarrow \texttt{Denoise}(\boldsymbol{\tau}_k, \hat{\boldsymbol{\epsilon}})$
    $\boldsymbol{\tau}_{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_{k-1}, \alpha\boldsymbol{\Sigma}_{k-1})$
end for
Extract prediction $(\boldsymbol{o}_C, \boldsymbol{o}_{C+1}, \ldots, \boldsymbol{o}_{C+H-1})$ from $\boldsymbol{\tau}_0$
```

B ILLUSTRATION OF MULTI-AGENT TRAJECTORY MODELING

To provide better understanding of how multiple agents' observations are modeled by MADiff in centralized control and decentralized execution scenarios, we show illustrative examples in a typical three-agent environment in Fig. 4. If the environment allows for centralized control, we can condition MADiff on all agents' historical and current observations, and let the model sample all agents' future trajectories as a single sample, as shown in Fig. 4(a). Then the current and next observations are sent to the inverse dynamics model for action prediction. If only decentralized execution are permitted, as shown in Fig. 4(b), agent 1 can only condition the model on its own information. The historical and current observations of other agents are masked when performing conditioning. MADiff now not only generates agent 1's own future trajectories, but also predicts the current and future observations of other two agents. Due to the joint modeling of all agents during training, such predictions are also reasonable and can be considered as a form of opponent modeling from agent 1's perspective. Although opponent modeling is not directly used in generating agent 1's ego actions, it can help agent 1 refine its planned trajectories to be consistent with predictions of others.


History Horizon
Horizon
Agent 0
Agent 1
Agent 2
o_0^0 o_1^0 o_2^0 o_3^0 o_4^0
o_0^1 o_1^1 o_2^1 o_3^1 o_4^1
o_0^2 o_1^2 o_2^2 o_3^2 o_4^2
Current Time step
Time steps
o_t^i : Masked Observations
o_t^i : Conditioned Observations
o_t^i : Generated Observations

(a) MADiff in centralized control.

(b) MADiff in decentralized execution.

Figure 4: Illustration of how agents' observations are modelled by MADiff in a three-agent environment. Note that figure (b) shows the situation when Agent 1 is taking action during decentralized execution.

C IMPLEMENTATION DETAILS

C.1 SCORE NORMALIZATION

The average scores of MPE tasks in Tab. 1 are normalized by the expert and random scores on each task. Denote the original episodic return as S, then the normalized score S_{norm} is computed as

$$S_{\text{norm}} = 100 \times (S - S_{\text{random}})/(S_{\text{expert}} - S_{\text{random}}) \,,$$

which follows Pan et al. (2022) and Fu et al. (2020). The expert and random scores on Spread, Tag, and World are {516.8, 159.8}, {185.6, -4.1}, and {79.5, -6.8}, respectively.

C.2 DETAILED NETWORK ARCHITECTURE

In Fig. 5, we unroll the U-Net structure of different agents in Fig. 1.

We describe the computation steps of attention among agents in formal. Each agent's local embedding c^i is passed through the key, value, and query network to form q^i, k^i, and v^i, respectively. Then the dot product with scaling is performed between all agents' q^i and k^i, which is followed by a Softmax operation to obtain the attention weight α^{ij}. Each α^{ij} can be viewed as the importance of j-th agent to the i-th agent at the current time step. The second dot product is carried out between the weight matrix and the value embedding v^i to get $\hat{c}^i$ after multi-agent feature interactions. Then $\hat{c}^i$ is skip-connected to the corresponding decoder block. The step-by-step computation of multi-agent

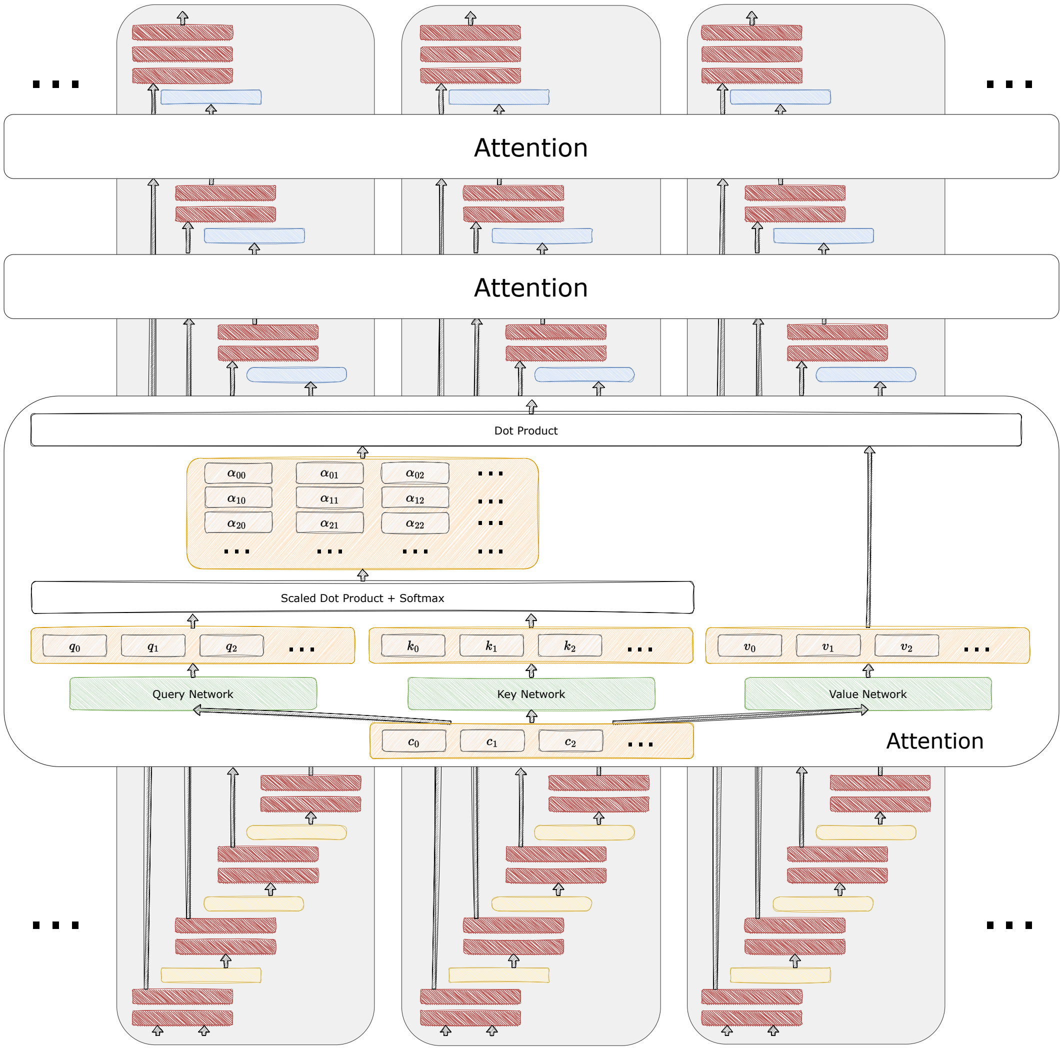


Figure 5: The detailed architecture of MADIFF. Each agent's U-Net is unrolled and lined up in the horizontal direction.

attention in MADIFF can be written as

$$
\begin{aligned}
q^i = f_{\text{query}}(c^i),\ k^i = f_{\text{key}}(c^i),\ v^i = f_{\text{value}}(c^i)\ ; \\
\alpha^{ij} = \frac{\exp(q^i k^j / \sqrt{d_k})}{\sum_{p=1}^{N} \exp(q^i k^p / \sqrt{d_k})}\ ; \\
\hat{c}^i = \sum_{j=1}^{N} \alpha^{ij} v^j\ ,
\end{aligned}
$$

where d_k is the dimension of k^i.

C.3 HYPERPARAMETERS

We list the key hyperparameters of MADIFF we used in Tab. 3 and Tab. 4. Return scale is the normalization factor used to divide the conditioned return before input to the diffusion model. The rough range of the return scale can be determined by the return distributions of the training dataset. We only tune the guidance weight ω, return scale, planning horizon H, and history horizon. We tried the guidance weight of $\{1.0, 1.2, 1.4, 1.6\}$, and found that different choices do not significantly affect final performances, in result we choose 1.2 as our default. We found larger planning horizon, in general, results in better performance. For MPE tasks, we find it unnecessary to condition on history observation sequence; thus, we set all history horizons to zero. In SMAC tasks which are more complex and more partially observable, we found a history horizon of 8, which is smaller than the planning horizon of 20, performs well across all datasets.

Table 3: Hyperparameters of MADIFF on MPE datasets.

TestBed	Spread				Tag				World			
Dataset	Expert	Md-Replay	Medium	Random	Expert	Md-Replay	Medium	Random	Expert	Md-Replay	Medium	Random
Return scale	700	500			700	600		500	700	600		
Learning rate	2e-4											
Guidance scale ω	1.2											
Planning horizon H	24											
History horizon	0											
Batch size	32											
Diffusion steps K	200											
Optimizer	Adam Optimizer											

Table 4: Hyperparameters of MADIFF on SMAC datasets.

TestBed	3m			5m6m		
Dataset	Good	Medium	Poor	Good	Medium	Poor
Return scale	20		10	20		10
Learning rate	2e-4					
Guidance scale ω	1.2					
Planning horizon H	20					
History horizon	8					
Batch size	32					
Diffusion steps K	200					
Optimizer	Adam Optimizer					

C.4 COMPUTING RESOURCES AND WALL TIME

We provide a concrete example as a reference for the time and resources required for training MADIFF. On a server with an AMD Ryzen 9 5900X (12 cores) CPU and an RTX 3090 GPU, we trained the MADIFF-C model on the Expert dataset from the MPE Spread task, which converged in about an hour. The curve of Wall-clock time spent for training and the corresponding model performance is shown in Fig. 6.

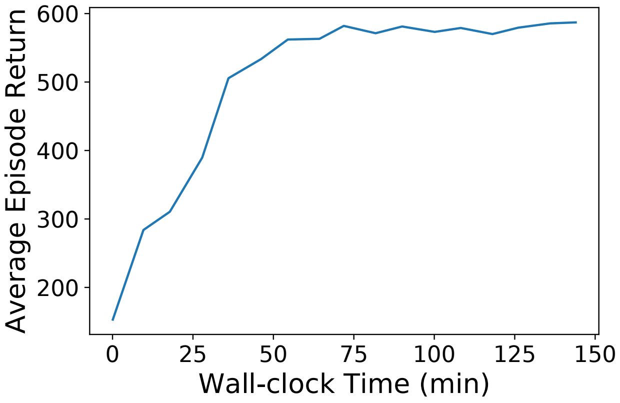


Figure 6: Wall-clock time and corresponding average episode return (average over 10 episodes) during training MADIFF-C for MPE Spread task.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 EFFECTIVENESS OF OPPONENT MODELING

To investigate whether opponent modeling can lead to performance improvements during decentralized execution, we conduct ablation experiments on MPE Spread datasets. We compare MADIFF-D with its variant that adopts the same network architecture but masks the diffusion loss on other agents' trajectories during training. We denote the variant as MADIFF-D w/o OM. The results are presented in Tab. 5, which show that opponent modeling results in notable performance improvements on all four levels of datasets.

Table 5: Ablation results of opponent modeling on MPE Spread datasets across 3 seeds.

Dataset	**MADIFF-D w/o OM**	**MADIFF-D**
Expert	93.4 ± 3.6	**98.4 ± 12.7**
Medium	35.4 ± 6.6	**53.2 ± 2.3**
Md-Replay	17.7 ± 4.3	**42.9 ± 11.6**
Random	5.7 ± 3.1	**19.4 ± 2.9**

D.2 OPPONENT MODELING ON SMAC TASKS

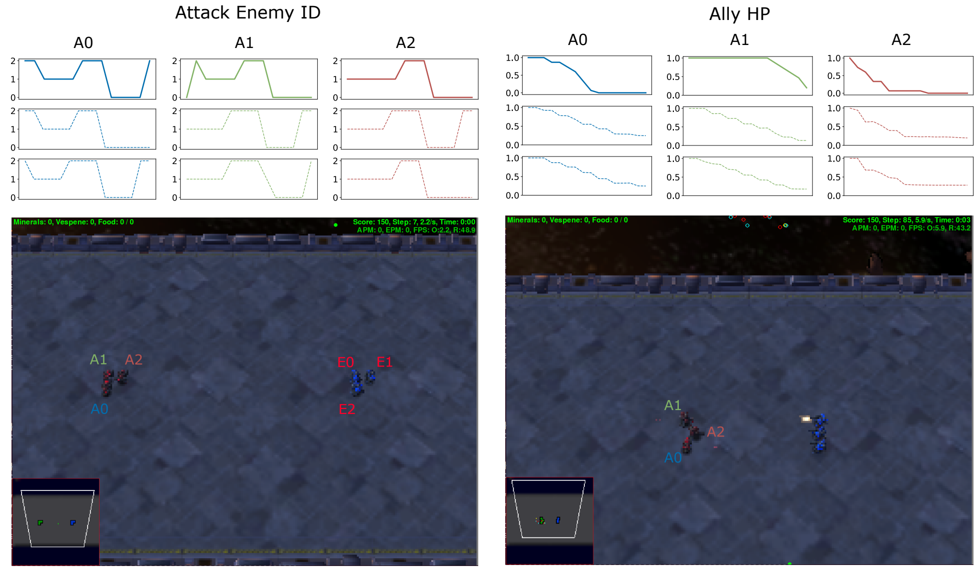
We show and analyze the quality of opponent modeling by MADIFF-D on SMAC. Specifically, we choose two-time steps from an episode on 3m map to analyze predictions on allies' attack targets and health points (HP), respectively.

On top of Fig. 7(a) is attacked enemy agent ID (0, 1, 2 stands for E0, E1, E2) of ally agents A0, A1, and A2. The first row is the ground-truth ID, and the second and the third rows are the predictions made by MADIFF-D from the other two allies' views. We can see that the predictions are in general consistent with the ground-truth ID. As can be seen from the true values of the attack enemy ID, agents tend to focus their firepower on the same enemies at the same time. And the accurate prediction of allies' attack enemy IDs intuitively can help to execute such a strategy.

In Fig. 7(b) we visualize the HP change curve of ally agents starting from another time step. From the environment state visualization below, agent A2 is the closest to enemies, so its HP drops the fastest. Such a pattern is successfully predicted by the other two agents.

D.3 PREDICTED TRAJECTORY VISUALIZATION ON NBA DATASET

We visualize the players' moving trajectories predicted by MADIFF-C and Baller2Vec++ on the NBA dataset in Fig. 8. In each image, the solid lines are real trajectories and the dashed lines are trajectories predicted by the model. The trajectories predicted by MADiff-C are closer to the real trajectories and are overall smoother compared to the Baller2Vec++ predictions.



(a) Ground-truth and predicted enemy's ID to attack by each ally agent.

(b) Ground-truth and predicted health points (HP) of each ally agent.

Figure 7: The ground-truth and predicted information of different MADIFF agents at two-time step. On the top of each figure, each column describes a different agent. The first row shows the change curve of the real value, and the last two rows below are the information predicted by other agents.

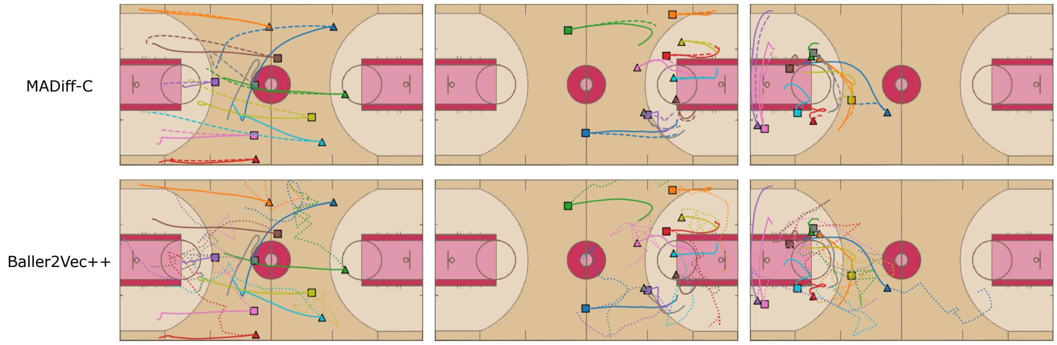


Figure 8: Real and Predicted multi-player trajectories by MADIFF-C and Baller2Vec++.