A APPENDIX

A.1 NETWORK ARCHITECTURE

In Fig.(6), we illustrate the detail of the network architecture for CMT with the input and output structure. The input consists of a sequence of trajectory tokens, which are embedded by a linear layer and add up with the position embedding. The output is decoded from the latent states in the transformer by another linear layer. Noticed that the output from history trajectory tokens is masked to avoid participating in the supervised loss.

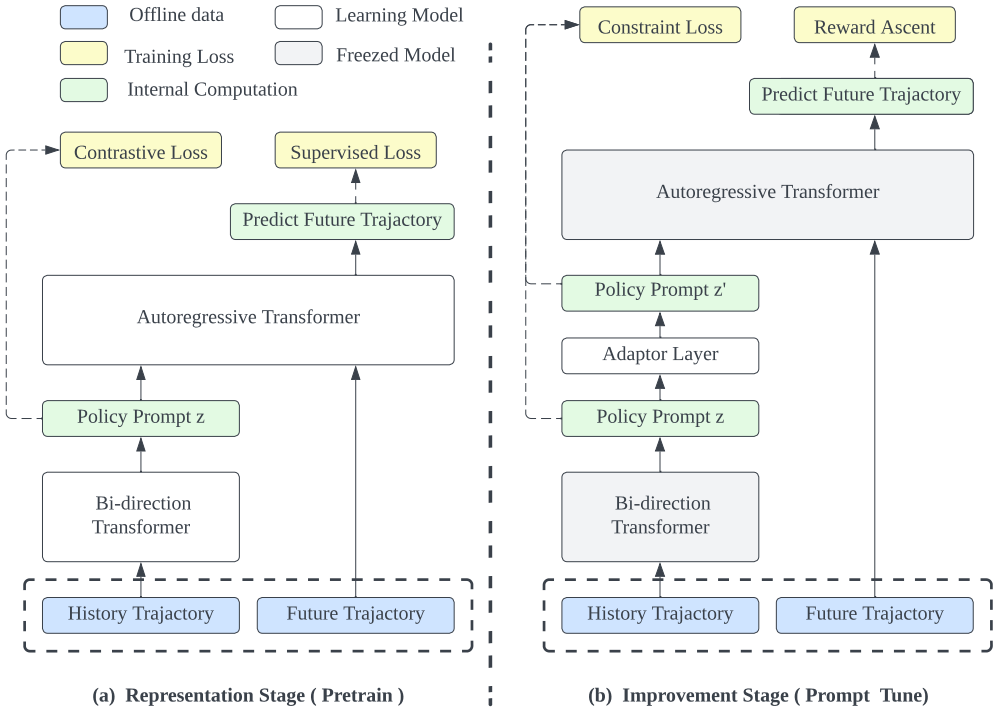

Figure 6: Detailed data flow, loss and network architecture for CMT.

A.2 HYPER-PARAMETER

In this section, we describe detailed hyperparameters to reproduce the experimental results. Due to the robustness of CMT, our algorithm shares similar hyperparameters among three benchmarks as shown in Table.(2).

In Table.(3), we discuss the distinct hyperparameters for four meta Mujoco tasks.

A.3 MULTI-AGENT OFFLINE LEARNING TASKS

By simply representing states and actions from several agents as a sequence of tokens, CMT can be deployed in the multi-agent tasks. In this subection, we evaluate the performance of CMT on multi-agent offline learning settings in SMAC benchmarks in 20 maps. For the data collection, we follow the same method in literature in (Meng et al., 2021). The datasets are built from trajectories generated by MAPPO on the SMAC tasks, and a large number of trajectories are contained in each of them. Different from D4RL, the properties of the DecPOMDP, the local observations and available actions, are also considered in our datasets.

The BC (Torabi et al., 2018), CQL-MA (Kumar et al., 2020), and ICQ-MA (Yang et al., 2021) are utilized as baselines to show the performance of our solution, and their original models own good performances in single-agent offline RL tasks. The properties of the multi-agent versions are the same as the single-agent versions. BC learns by building the state-to-action mapping. Based on the traditional multi-agent offline RL methods, ICQ-MA and CQL-MA solve the extrapolation error problem through action-space constraint and value pessimism, respectively.

The results on eight maps are displayed in Fig. 7 to demonstrate the performance of algorithms on tasks of varying difficulty (Super hard: $MMM2$, $corridor$, $3s5z\_vs\_3s6z$; Hard: $3s\_vs\_5z$, $8m\_vs\_9m$, $3s5z$; Easy: $8m$, $3s\_vs\_4z$). More results on StarCraft II can be found in Appendix. The CMT outperforms the baselines and achieves state-of-the-art performance in all maps, indicating that our algorithm has strong robustness and high efficiency. While ICQ-MA and CQL-MA perform poorly due to extrapolation errors and larger errors generated by multiple agents. Furthermore, it should be noted that the BC works well since the approximate expert datasets are used in training stage.

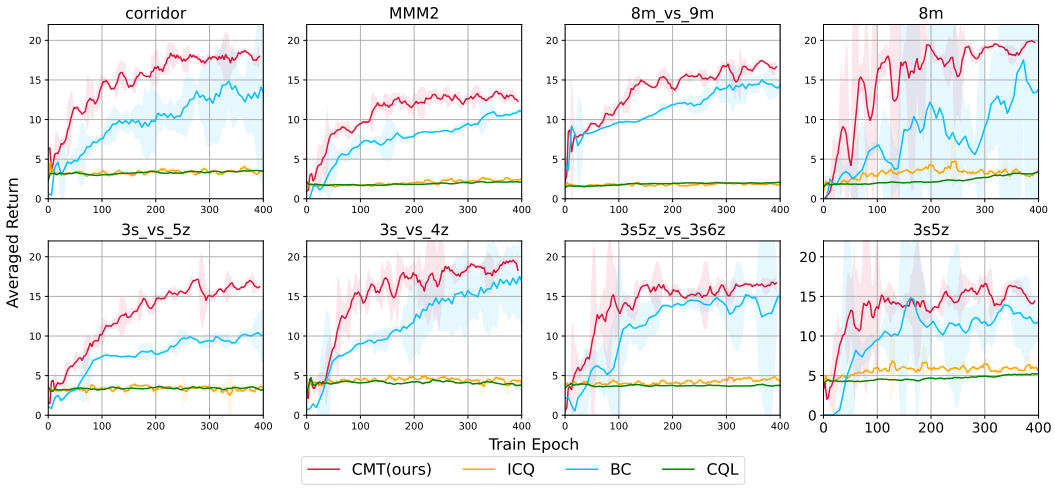

Figure 7: Results for eight representative maps in SMAC. CMT has significant advantages, compared with ICQ, CQL, BC baselines. All results on 20 maps can be found in the appendix.

A.4 Full resluts on SMAC

We evaluate CMT on twenty maps in the SMAC benchmark. As shown in Fig.(8), the results demonstrate that CMT remarkably outperforms baselines, including BC, ICQ, and CQL.

Parameter	D4RL(Default Config)	meta Mujoco	SMAC
Optimizer	AdamW	AdamW	AdamW
Batch size	256	512	256
learning rate	1e-4	1e-4	1e-4
Transformer block layer	2	2	2
Attention head	2	2	2
Embedding dimension	32	32	32
context length - policy	40	30	10
context length - task	None	30	None
gradient norm clip	0.5	0.5	0.5
contrastive loss - K	256	512	256
contrastive loss - α	0.2	0.2	0.2
contrastive loss - γ	0.1	0.1	0.1
behavioral constraint - β	1	1	1

Table 2: Common hyper-parameters for CMT in D4RL, meta Mujoco and SMAC.

Parameter	Ant-Fwd-Bwd	Half-CHeetah-Fwd-Bwd	Point-Robot-Wind	Walker-2D-Params
train tasks number	2	2	40	40
test task number	2	2	10	10
context length -task	32	64	32	32

Table 3: Specfic hyper-parameters for four mete Mujoco tasks.

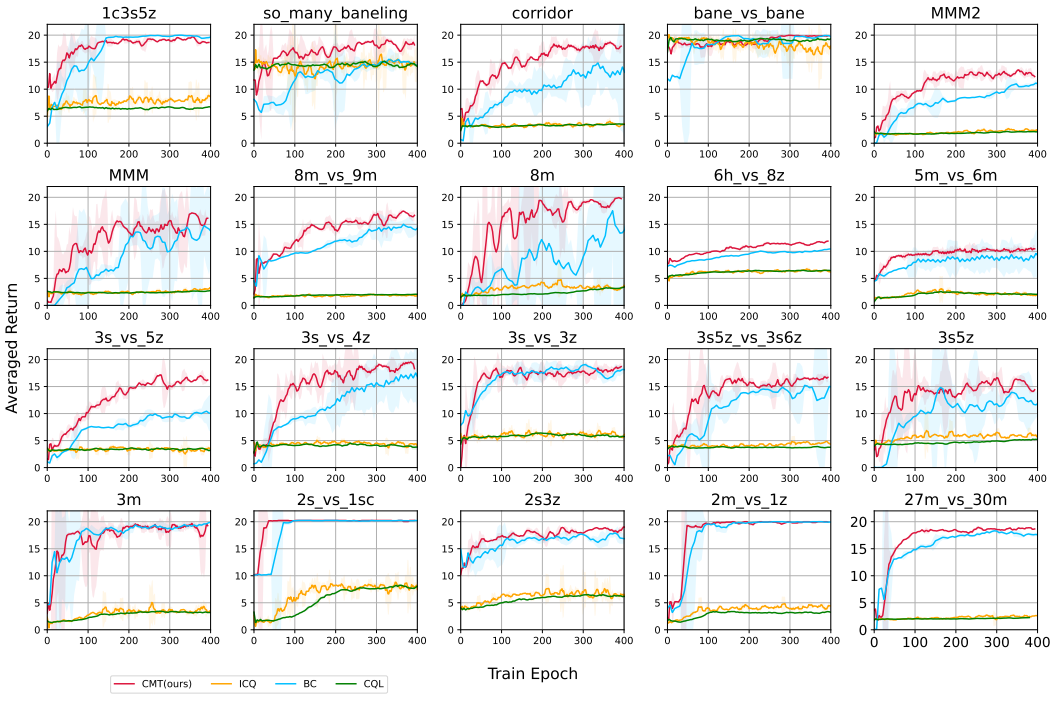

Figure 8: Results for SMAC on twenty maps.CMT has significant advantages, compared with ICQ, CQL, BC baselines.