7 APPENDIX

7.1 Architecture

For tasks without visual observations, we directly adopt the deep neural network as the actor network as well as the critic network. Below is the architecture of the deep neural network.

```
# The architecture of the actor network:

nn.Sequential(
    nn.Linear(n_o, 128),
    nn.ReLU(),
    nn.Linear(128, 64),
    nn.ReLU(),
    nn.Linear(64, 32),
    nn.ReLU(),
    nn.Linear(32, 16),
    nn.ReLU(),
    nn.Linear(16, n_a)
)

# The architecture of the critic network:

nn.Sequential(
    nn.Linear(n_o, 128),
    nn.ReLU(),
    nn.Linear(128, 64),
    nn.ReLU(),
    nn.Linear(64, 32),
    nn.ReLU(),
    nn.Linear(32, 16),
    nn.ReLU(),
    nn.Linear(16, 1)
)
```

where n_o is the dimension of the observation and n_a represents the number of the optional actions for tasks with discrete action space or the dimension of the action for tasks with continuous action space.

For tasks with visual observations, we perform two convolutional layers, self.conv1 and self.conv2, on the original observation, flatten the output by self.conv2, and then obtain the reshaped observation. After this, we perform the same operation as tasks without visual observations.

```
# in_channel, out_channel, kennel_size, stride
self.conv1 = nn.Conv2d(3, 1, (3, 4), (12,16))
self.conv2 = nn.Conv2d(1, 1, (3, 4), (3,4))
```

7.2 Results visualization

We plot the performance of our selected baselines for gym-minigrid, MT10, MT50, Atari, Ravens, and RLBench in Figures 6, 7, 9, 10, 11, and 12.

7.3 Convergence of Vanilla Frank-Wolfe Algorithm

7.3.1 Experimental Analysis

We randomly select 10 moments of performing the Vanilla Frank-Wolfe algorithm when optimizing b_t^o and visualize the optimization process in Figure 5. The evaluation metric is the objective function in Eq. (4) with the task t fixed.

7.3.2 Theoretical Analysis

Corollary. *Denote x_m as the solution generated by vanilla Frank–Wolfe at the m-th round. Then*

$$\min_{1\le j\le m} s\ge 0, \|s\|_1 \le radius \langle \nabla f(x_m), x_m - s\rangle \tag{8}$$
$$\le \frac{\max\{2h_0, 4\{\lambda_0(\mu_1+\mu_2)D_1+\lambda_1 D_2^2+2(\lambda_2+\lambda_3+\lambda_4)dT\}T\}}{\sqrt{m+1}},$$

where $h_0 = f(x_0) - \min_{x\ge 0, \|x\|_1\le radius} f(x)$, D_1 is the upper bound of $\{|\mathcal{L}(w_i)|\}_{i\in[T]}$, and D_2 is the 2-norm upper bound of the neural network weights. Note that, in our experiments, $D_2 \le 20$.

When f is convex and differentiable,

$$\min_{1\le j\le m} s\ge 0, \|s\|_1 \le radius \langle \nabla f(x_m), x_m - s\rangle$$
$$\le \frac{8\{\lambda_0(\mu_1+\mu_2)D_1+\lambda_1 D_2^2+2(\lambda_2+\lambda_3+\lambda_4)dT\}T}{m+1}. \tag{9}$$

To sum up, the convergence rate of vanilla Frank–Wolfe on problem Eq. (7) is between $\frac{1}{m}$ and $\frac{1}{\sqrt{m}}$.

Remark. $\max_{s\ge 0, \|s\|_1\le radius} \langle \nabla f(x_m), x_m - s\rangle$ *is the standard evaluation metric of vanilla Frank–Wolfe.*

Proof. Recall that $f(b_t^o) = \lambda_0[(1+\mu_1\sum_{j\in[T]\setminus\{t\}} B_{tj})\mathcal{L}(w_t) - \mu_2(b_t^o)^\top l_t^o] + \lambda_1\sum_{s\in\mathcal{U}\setminus t}\|w_s - \sum_{j=1}^{i-1} B_{\pi(j)s}w_{\pi(j)} - B_{ts}w_t\|_2^2 + \lambda_2\sum_{j\in[q]}(j - y\prime_{i_j})^2 + \lambda_3\sum_{j\in[T]}(rank_j^{(1)} - y\prime\prime_j)^2 + \lambda_4\sum_{j\in[T]}(rank_j^{(2)} - y\prime\prime_j)^2$.

Among the terms in $f(b_t^o)$, $(1+\mu_1\sum_{j\in[T]\setminus\{t\}} B_{tj})\mathcal{L}(w_t) - \mu_2(b_t^o)^\top l_t^o$ is $(\mu_1+\mu_2)D_1T$-Lipschitz with respect to b_t^o and $\lambda_1\sum_{s\in\mathcal{U}\setminus t}\|w_s - \sum_{j=1}^{i-1} B_{\pi(j)s}w_{\pi(j)} - B_{ts}w_t\|_2^2$ is $\lambda_1 D_2^2 T$-Lipschitz with respect to b_t^o, respectively.

According to $[\tanh(x)]' = 1 - (\tanh(x))^2$, it can be easily deduced that $\lambda_2\sum_{j\in[q]}(j - y\prime_{i_j})^2 + \lambda_3\sum_{j\in[T]}(rank_j - y\prime\prime_j)^2 + \lambda_4\sum_{j\in[T]}(rank_j^{(2)} - y\prime\prime_j)^2$ is $2(\lambda_2+\lambda_3+\lambda_4)dT^2$-Lipschitz with respect to b_t^o.

By integrating the above terms, we obtain the $\{\lambda_0(\mu_1+\mu_2)D_1 + \lambda_1 D_2^2 + 2(\lambda_2+\lambda_3+\lambda_4)dT\}T$-Lipschitz property of $f(b_t^o)$. Also note that the diameter of b_t^o's feasible solution space is no greater than 1, so by Equation 3 in [18] and Equation 3.11 in [23] we can easily deduce the two inequalities in Eq. (8) and Eq. (9). □

TABLE 7: Reward before and after the incorporation of prior knowledge for MT10.

	Ranking of performance by soft-module [32]	CAMRL w/o prior success rate	CAMRL w/ prior success rate
Pick and place	9	**47235.31**	42798.2
Pushing	8	213.27	**227.94**
Reaching	2	**-50.36**	-51.28
Door opening	6	143.74	**147.64**
Button press	4	1395.77	**1449.87**
Peg insertion side	10	**52.91**	50.03
Window opening	3	**15683.29**	14948.5
Window closing	7	23.64	**26.97**
Drawer opening	5	1329.39	**1403.84**
Drawer closing	1	**622.74**	618.76

TABLE 8: Reward before and after the incorporation of prior knowledge for MT50.

	Ranking of performance by MT-SAC of meta-world benchmark [36]	CAMRL w/o prior success rate	CAMRL w/ prior success rate
Turn on faucet	1	7644.89	**7928.39**
Sweep	37	-10.31	**-6.38**
Stack	NA (not found)	**-42.32**	-43.21
Unstack	NA (not found)	-31.26	**-30.48**
Turn off faucet	1	1253.73	**1308.41**
Push back	34	-27.74	**-26.37**
Pull lever	36	-31.84	**-30.37**
Turn dial	1	**-27.69**	-28.36
Push with stick	37	1376.83	**1427.37**
Get coffee	31	-52.93	**-49.31**
Pull handle side	1	**-48.93**	-49.76
Basketball	37	16394.92	**17389.21**
Pull with stick	37	-28.92	**-26.43**
Sweep into hole	NA (not found)	16.74	**17.38**
Disassemble nut	37	**1873.93**	1793.35
Place onto shell	NA (not found)	274.01	**304.74**
Push mug	NA (not found)	638.91	**652.48**
Press handle side	1	24.18	**25.31**
Hammer	26	-60.93	**-58.42**
Slide plate	1	425.64	**445.31**
Slide plate side	24	**-23.54**	-24.01
Press button wall	1	-27.73	**-26.46**
Press handle	1	-49.18	**-47.54**
Pull handle	1	46.84	**47.14**
Soccer	31	**485.03**	472.77
Retrieve plate side	1	127.94	**130.58**
Retrieve plate	1	**-42.65**	-44.51
Close drawer	1	428.93	**441.93**
Press button top	1	-29.84	**-24.18**
Reach	1	-23.47	**-20.45**
Press button top w/wall	1	-41.84	**-38.53**
Reach with wall	1	897.42	**910.58**
Insert peg side	25	**12.93**	10.56
Push	30	**573.94**	570.53
Push with wall	35	23.81	**30.51**
Pick & place w/wall	37	-33.27	**-30.56**
Press button	1	847.93	**856.25**
Pick & place	37	-23.15	**-21.16**
Pull mug	NA (not found)	-3.03	**-2.74**
Unplug peg	29	-52.94	**-48.61**
Close window	1	-21.19	**-20.68**
Open window	22	**-27.36**	-29.63
Open door	22	-62.27	**-58.64**
Close door	1	**-23.01**	-25.96
Open drawer	27	-60.03	**-56.61**
Open box	33	-27.09	**-24.45**
Close box	28	-47.15	**-44.58**
Lock door	1	-45.39	**-40.99**
Unlock door	1	**-33.06**	-36.81
Pick bin	37	-42.94	**-40.51**

TABLE 9: Reward before and after the incorporation of prior knowledge for Atari.

	Ranking of performance in the Atari Learderboard	CAMRL w/o prior reward	CAMRL w/ prior reward
YarsRevenge	1	1637.94	**1684.8**
Jamesbond	NA (not found in the Atari leaderboard [26])	**8.23**	7.96
FishingDerby	7	-11.41	**-7.43**
Venture	5	0.22	**0.24**
DoubleDunk	NA (not found)	-0.89	**-0.75**
Kangaroo	3	5.74	**5.94**
IceHockey	8	**-0.49**	-0.52
ChopperCommand	2	351.68	**374.95**
Krull	4	**9.75**	9.49
Robotank	6	0.38	**0.42**

TABLE 10: Success rate before and after the incorporation of prior knowledge for Ravens.

	Ranking of performance by Transporter network [37]	CAMRL w/o prior success rate	CAMRL w/ prior success rate
Block-insertion	2	94.26	**94.78**
Place-red-in-green	1	93.44	**93.73**
Towers-of-hanoi	4	95.12	**95.34**
Align-box-corner	3	93.43	**94.01**
Stack-block-pyramid	10	**77.15**	74.24
Palletizing-boxes	5	94.32	**94.68**
Assembling-kits	7	93.95	**93.99**
Packing-boxes	9	**79.38**	79.06
Manipulating-rope	8	89.27	**90.24**
Sweeping-piles	6	93.64	**93.87**

TABLE 11: Success rate before and after the incorporation of prior knowledge for RLBench.

	Ranking of performance by [21]	CAMRL w/o prior success rate	CAMRL w/ prior success rate
Reach Target	1	99.87	**99.93**
Push Button	2	97.24	**98.02**
Pick And Lift	3	91.58	**92.04**
Pick Up Cup	4	86.48	**86.53**
Put Knife on Chopping Board	9	50.91	**55.46**
Take Money Out Safe	8	**66.84**	66.79
Put Money In Safe	6	80.38	**82.44**
Pick Up Umbrella	5	81.49	**82.76**
Stack Wine	10	24.59	**26.98**
Slide Block To Target	7	79.47	**81.46**

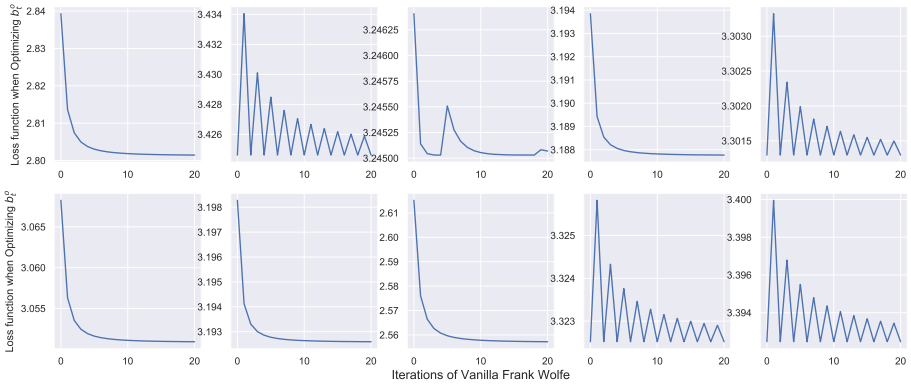

Fig. 5: Convergence of the vanilla Frank-Wolfe algorithm to optimize b_t^o.

TABLE 12: Ablation study. Here '-' means 'without', and '+' means 'CAMRL with a specific setting'. a, b, and c are the coefficients of the indicator I_{mul}. $\lambda_i (i \in [4])$ are coefficients of each term (except the first term) in Eq. (6).

Method \ Reward	Pick and place	Pushing	Reaching	Door opening	Button press	Peg insertion side	Window opening	Window closing	Drawer opening	and Drawer closing	Avg. reward
CAMRL	**45052.2**	204.3	-53.3	**138.5**	**1382.4**	**53.3**	**6764.8**	10.3	**1321.7**	**1512.9**	**5638.7**
AMRL	10841.9	10.1	-62.2	30.4	984.2	2.9	5481.7	**21.1**	104.6	1033.8	1844.9
- mode switch	18849.3	19.7	**-50.4**	80.5	1163.8	30.6	5791.4	8.9	498.8	1402.1	2779.5
+ a=positive infinity	35318.4	170.4	-55.3	125.5	1303.8	44.1	6191.7	9.2	1184.2	1421.4	4571.3
+ b=positive infinity	35024.8	149.8	-58.9	104.8	1284.3	41.7	5938.1	9.1	872.9	1284.5	4465.1
+ c=0	38529.1	192.6	-53.8	134.9	1389.4	47.9	6268.8	10.6	1288.7	1482.9	4929.1
+ d=0	23029.4	21.8	-61.9	94.7	1228.6	36.1	5893.3	8.4	841.2	1465.5	3255.7
+ λ_2=0	36931.3	109.4	-57.7	112.9	1255.7	42.6	4193.5	14.9	1201.8	1449.2	4525.4
+ λ_3=0	43093.8	**215.4**	-56.3	129.4	1372.5	47.7	6288.3	12.6	1279.1	1485.6	5386.8
+ λ_4=0	44226.7	202.5	-53.8	130.8	1368.4	48.1	6631.2	9.3	1249.5	1474.5	5528.7

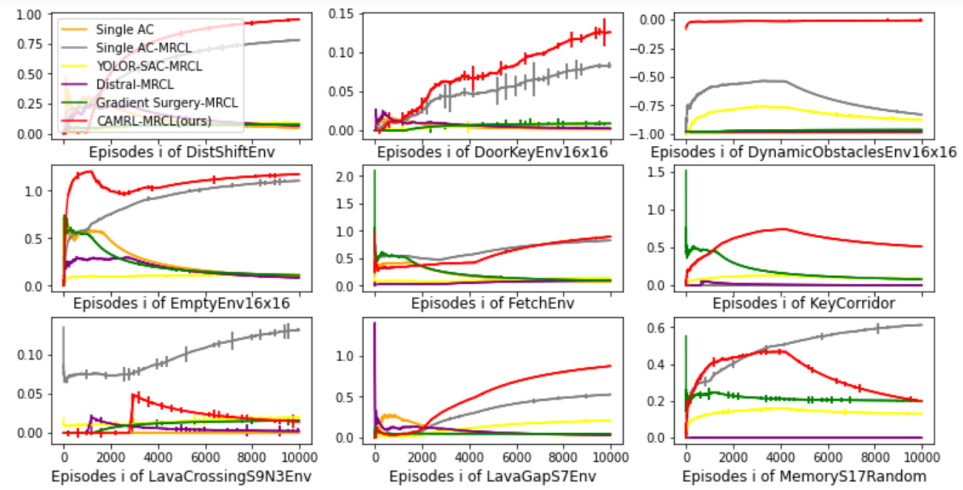

Fig. 6: Averaged reward curve of the first $10K$ episodes for Gym-minigrid tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.

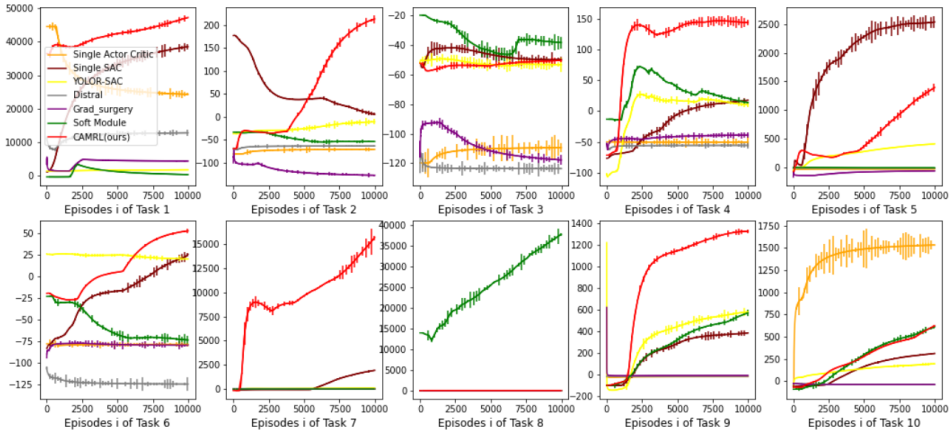

Fig. 7: Averaged reward curve of the first $10K$ episodes for MT10 tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.

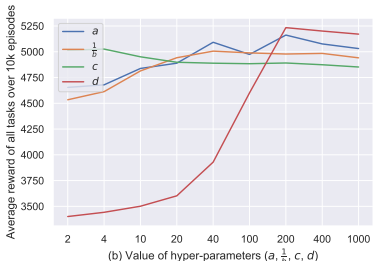

Fig. 8: Hyper-parameter analysis of CAMRL.

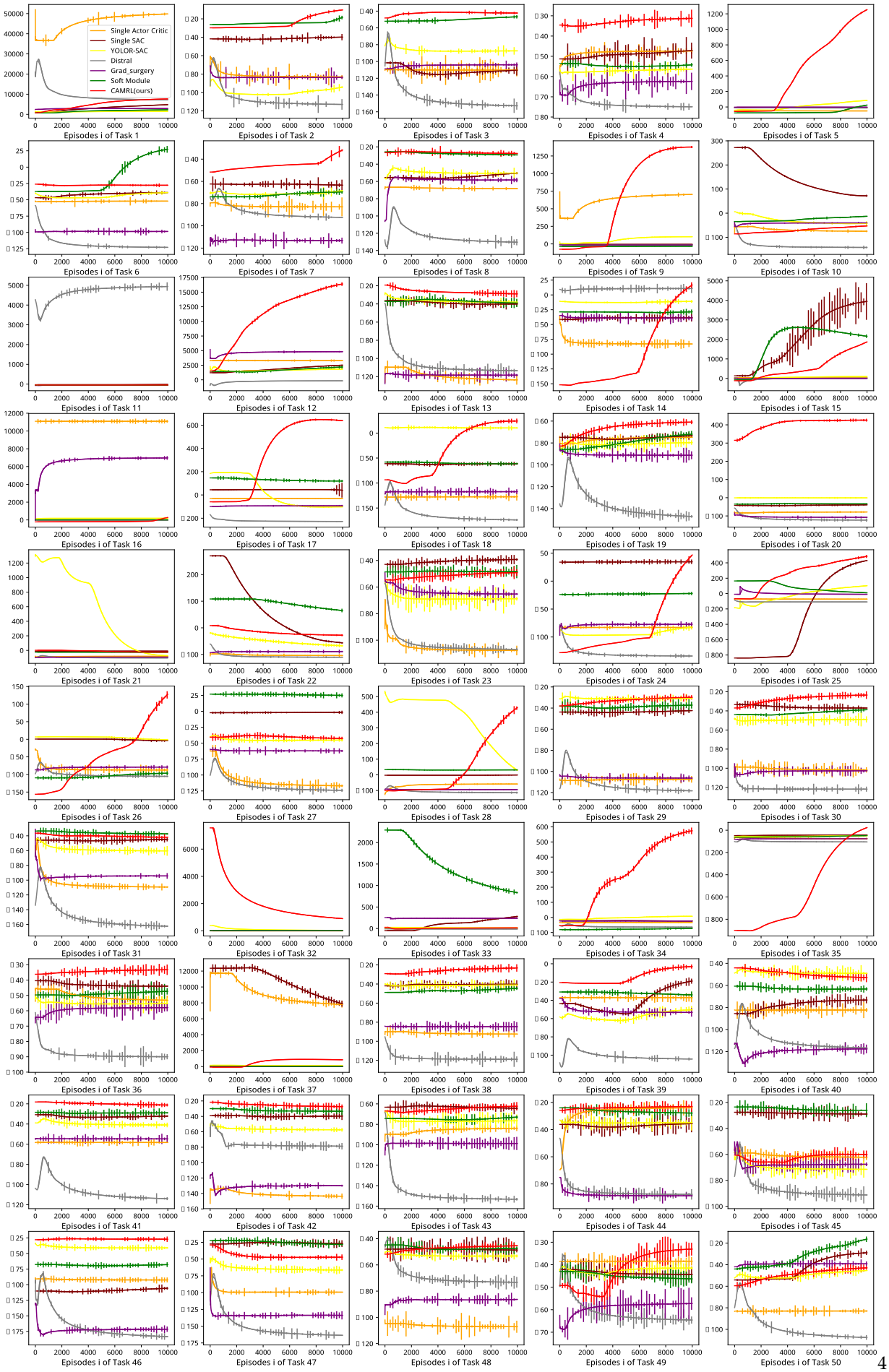

Fig. 9: Averaged reward curve of the first $10K$ episodes for MT50 tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.

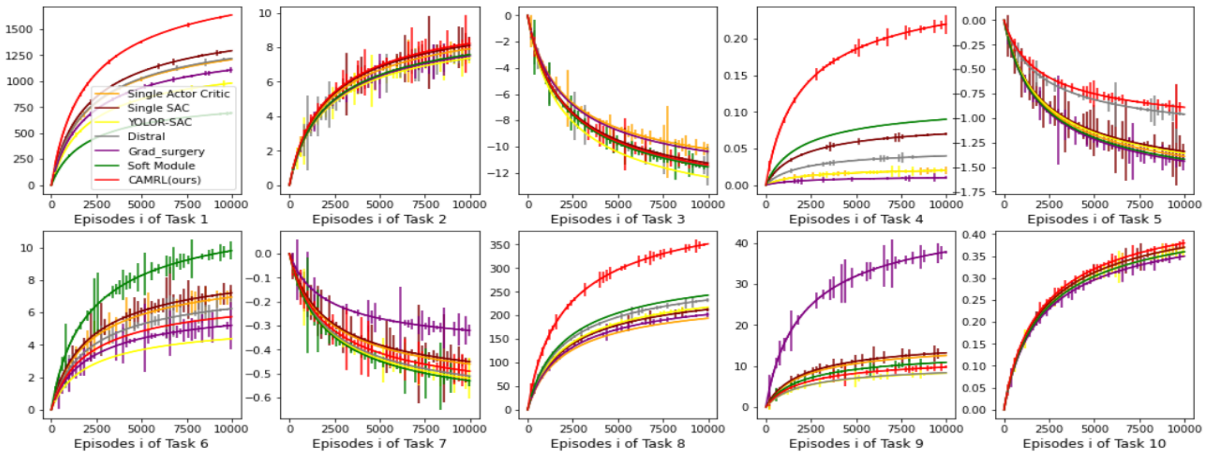

Fig. 10: Averaged reward curve of the first $10K$ episodes for Atari tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.

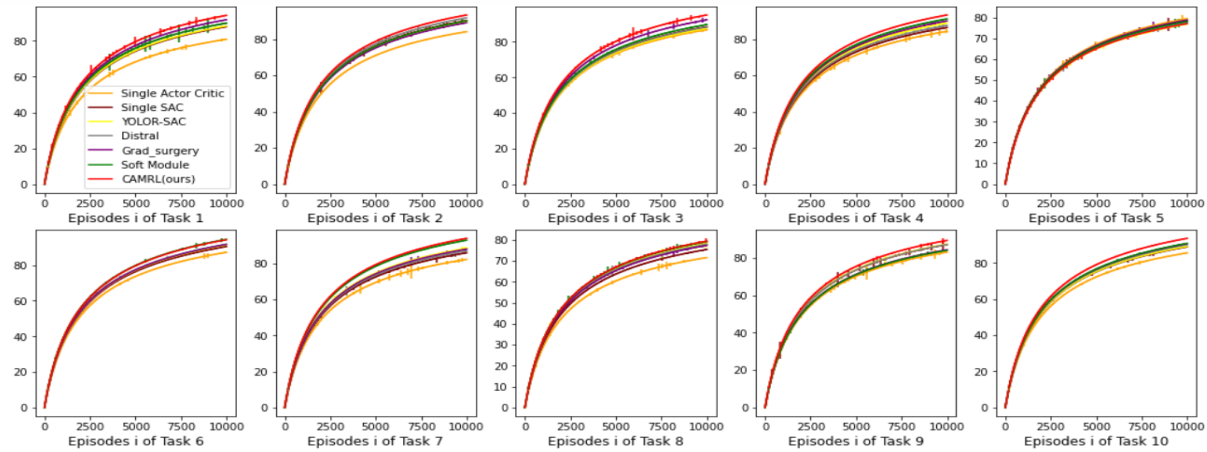

Fig. 11: Averaged reward curve of the first $10K$ episodes for Ravens tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.

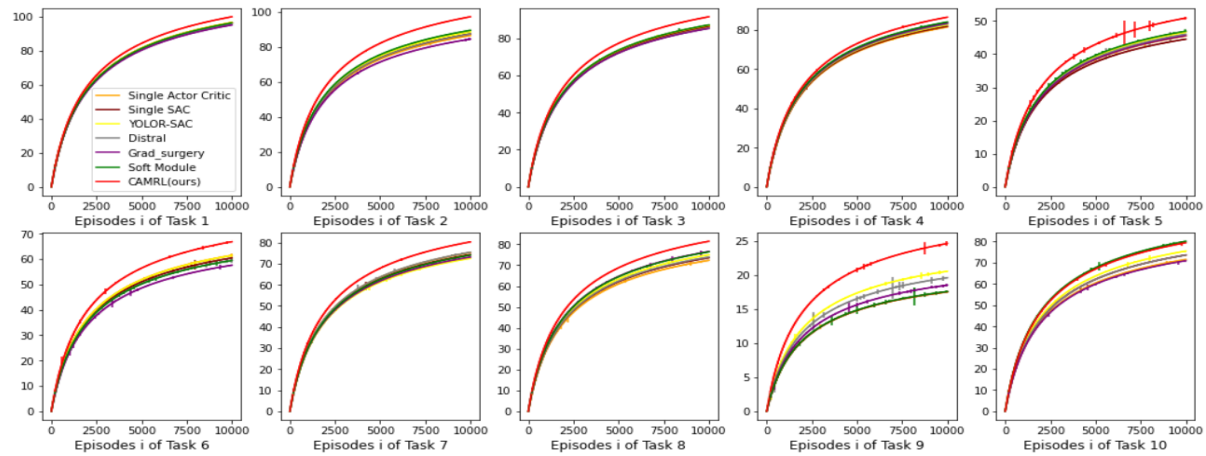

Fig. 12: Averaged reward curve of the first $10K$ episodes for RLBench tasks (each experiment repeated 5 times). The higher the metric, the better performance of the model.