A Proof

In this section, we provide the proof for the main theorem. We first restate the following assumptions.

Assumption A.1. *(Independency of Probing Data) The distribution of probing data is independent of the model, i.e.* $p(x) = p(x|M)$ *and* $p(M|x) = p(M)$.

Assumption A.2. *(Deterministic Model) For input* $x \in \mathcal{S} \times \mathcal{A}$ *and model* $M \in \mathcal{M}$, $p(y|x, M) = \delta(y = y^M)$ *where* $y^M = M(x)$.

Before proving the theorem, we introduce the following lemma.

Lemma A.1. *(Data Processing Inequality) Assume three random variables from the Markov Chain* $X \to Y \to Z$, *then* $I(X; Z) = I(X; Y) - I(X; Y|Z)$.

Proof. Applying the chain rule of mutual information, we have

$$\begin{aligned} I(X; Z) &= I(X; Y, Z) - I(X; Y|Z) \\ &= I(X; Z|Y) + I(X; Y) - I(X; Y|Z) \\ &= I(X; Y) - I(X; Y|Z), \end{aligned}$$

where the last equation holds for $I(X; Z|Y) = 0$ because of the Markov property. □

Then we move on to prove our main theorem.

Theorem A.2. *(Theorem 4.1 restated) Let* $x_{1:n}$ *denote i.i.d. probing data sampled from distribution* $\rho(x)$ *and the probing data with sampled models from* $p(M)$ *is labeled to construct* $y_{1:n}$. *With Assumptions* (A.1) *and* (A.2)*, optimizing* $\mathcal{J}(\theta, \psi)$ *in terms of the representation encoder* $q_\theta(x_{1:n}, y_{1:n})$ *corresponds to optimizing the lower bound of* $I(M; z)$*:*

$$I(M; z) \geq I(M; y_{1:n}|x_{1:n}) + \mathcal{J}(\theta, \psi).$$

Proof. To see this, we first note that probing the model M with data $x_{1:n}$ can be deemed as an implicit data-processing of the model, which implies the dependency graph in Figure S1.

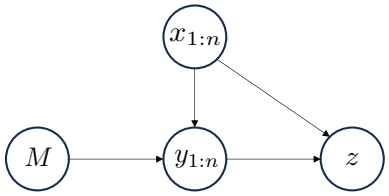

Figure S1: The dependency graph of the variables.

Then

$$\begin{aligned} &I(M; z) \\ &= H(M) - H(M|z) \\ &= H(M|x_{1:n}) - H(M|z, x_{1:n}) \\ &= I(M; z|x_{1:n}) \\ &= I(M; y_{1:n}|x_{1:n}) - I(M; y_{1:n}|z, x_{1:n}) \\ &= I(M; y_{1:n}|x_{1:n}) - H(y_{1:n}|z, x_{1:n}) + H(y_{1:n}|M, z, x_{1:n}) \\ &= I(M; y_{1:n}|x_{1:n}) - H(y_{1:n}|z, x_{1:n}), \end{aligned} \tag{1}$$

where the second equation holds due to Assumption (A.1), and the last equation holds due to Assumption (A.2).

Expanding $H(y_{1:n}|z, x_{1:n})$ leads to

$$\begin{aligned} &- H(y_{1:n}|z, x_{1:n}) \\ &= \mathbb{E}_{x_{1:n}} \mathbb{E}_{p(z, y_{1:n}|x_{1:n})} \left[\log p(y_{1:n}|z, x_{1:n}) \right]. \end{aligned} \tag{2}$$

However, it is intractable to estimate the posterior $p(y^M_{1:n}|z, x_{1:n})$, so we opt for a variational distribution $q_\psi(y^M_{1:n}|z, x_{1:n})$, and we have

$$\begin{aligned} &- H(y_{1:n}|z, x_{1:n}) \\ &= \mathbb{E}_{x_{1:n}} \mathbb{E}_{p(z, y_{1:n}|x_{1:n})} \Bigg[\log \frac{p(y_{1:n}|z, x_{1:n})}{q_\psi(y_{1:n}|z, x_{1:n})} \\ &\qquad + \log q_\psi(y_{1:n}|z, x_{1:n}) \Bigg] \\ &= \mathbb{E}_{x_{1:n}, z} \left[\mathrm{KL}(p \| q_\psi) \right] \\ &\qquad + \mathbb{E}_{x_{1:n}} \mathbb{E}_{p(z, y_{1:n}|x_{1:n})} \left[\log q_\psi(y_{1:n}|z, x_{1:n}) \right] \\ &\geq \mathbb{E}_{x_{1:n}} \mathbb{E}_{p(z, y_{1:n}|x_{1:n})} \left[\log q_\psi(y_{1:n}|z, x_{1:n}) \right], \end{aligned} \tag{3}$$

where the inequality is due to the non-negativity of KL divergence.

Finally we use the Assumption A.1 again to avoid the integral over $y_{1:n}$, and come to

$$\begin{aligned} &\mathbb{E}_{x_{1:n}} \mathbb{E}_{p(z, y_{1:n}|x_{1:n})} \left[\log q_\psi(y_{1:n}|z, x_{1:n}) \right] \\ &= \mathbb{E}_{x_{1:n}} [\int p(z, y_{1:n}) \log q_\psi(y_{1:n}|z, x_{1:n}) \mathrm{d}z \mathrm{d}y_{1:n}] \\ &= \mathbb{E}_{x_{1:n}} \mathbb{E}_M \mathbb{E}_{z \sim q_\theta(z|x_{1:n}, y^M_{1:n})} \left[\log q_\psi(y^M_{1:n}|z, x_{1:n}) \right] \\ &= \mathbb{E}_{x_{1:n}} \mathbb{E}_M \mathbb{E}_{z \sim q_\theta(z|x_{1:n}, y^M_{1:n})} \left[\sum_{k=1}^{n} \log q_\psi(y^M_k|z, x_k) \right] \\ &= \mathcal{J}(\theta, \psi), \end{aligned} \tag{4}$$

where the penultimate equation is due to the i.i.d. property of $x_{1:n}$.

Summarizing the above proof, we have

$$I(M; z) \geq I(M; y_{1:n}|x_{1:n}) + \mathcal{J}(\theta, \psi). \tag{5}$$

Thus with a fixed probing distribution, maximizing $\mathcal{J}(\theta, \psi)$ corresponds to maximizing the lower bound of $I(M; z)$. □

B Experimental Details

B.1 Meta-Environments

Here, we present the construction details of the meta-environments in the benchmark.

Configuration	Point-Robot	Cheetah-Dir	Cheetah-Vel	Ant-Dir	Hopper-Params	Walker-Params
Number of training tasks	10	2	10	10	10	10
Number of testing tasks	10	0	10	10	10	10
Trajectory number	100	100	50	50	50	50
Trajectory length	20	200	200	200	200	200
SAC training step	1e5	1e6	1e6	1e6	1e6	1e6

Table S1: Configurations used in dataset collection

Hyperparamter	Point-Robot	Cheetah-Dir	Cheetah-Vel	Ant-Dir	Hopper-Params	Walker-Params
Latent space dim	5	5	5	5	10	10
Task batch size	10	4	10	10	10	10
TAE batch size	256	256	256	256	256	256
RL batch size	256	256	256	256	256	256
TAE learning rate	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
RL learning rate	3e-4	3e-4	3e-4	3e-4	3e-4	3e-4
Discount factor	0.9	0.99	0.99	0.99	0.99	0.99
Training step	2e5	2e5	2e5	2e5	2e5	2e5
RL network width	64	256	256	256	256	256
RL network depth	4	4	4	4	4	4
TAE loss weight	10	10	10	10	10	10

Table S2: Configurations and hyperparameters used in meta-training

Point-Robot. Starting from the initial position randomly generated in the state space, the agent needs to navigate to the goal position. The goal position is uniformly sampled from $U[-1,1]\times U[-1,1]$. The reward function is dependent on the distance from the current position to the goal position.

Cheetah-Dir. It is a Half-cheetah environment with a target direction. The direction is either forward or backward. The reward function is dependent on the walking direction and the walking velocity.

Cheetah-Vel. It is a Half-cheetah environment with a target velocity which is uniformly sampled from $U[0,6]$. The reward function is dependent on how close the current velocity is to the goal velocity.

Ant-Dir. It is an Ant environment with a target direction which is uniformly sampled from $U[0,2\pi]$. The reward function is dependent on the velocity on the target direction.

Hopper-Params. It is a Hopper environment with different parameters. Transition function varies in randomized task-specific parameters like body mass, inertia, damping, and friction coefficients. Each parameter is sampled by multiplying the default value with $1.5^{\mu}, \mu \sim U[-3,3]$. The reward function is dependent on the forward velocity.

Walker-Params. It is a Walker environment with different parameters. Transition function varies in randomized task-specific parameters like body mass, inertia, damping, and friction coefficients. Each parameter is sampled by multiplying the default value with $1.5^{\mu}, \mu \sim U[-3,3]$. The reward function is dependent on the forward velocity.

For all the environments except for Cheetah-Dir, 10 training tasks and 10 testing tasks are randomly sampled. For Cheetah-Dir with only 2 tasks, we test the algorithm only on training tasks. For offline dataset generation, we train a SAC agent to the expert level on every single task, and roll out the expert-level agent to collect some trajectories as the offline datasets. The configurations in the data collection phase are listed in Table S1.

B.2 Baseline Algorithms

We re-implement FOCAL, CORRO, and BOReL with TD3+BC. TD3+BC updates policy with a behavior cloning term:

$$\max_{\pi} \mathbb{E}_{(s,a)\sim D}[\lambda Q(s,\pi(s)) - (\pi(s)-a)^2], \tag{6}$$

where

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s,a)}|Q(s,a)|}, \tag{7}$$

which balances the TD3's objective and the regularization term. Following the original paper, we set the hyperparameter α to 2.5 in all the environments, which shows reasonable performance in all baseline algorithms. We conduct the main experiments with the original baselines, in which FOCAL uses BRAC, and CORRO and BOReL use SAC as their backbone offline RL algorithms. We run FOCAL, CORRO, and BOReL with the open-source implementations for a fair comparison, and the results are listed in Table S3. The performances of the original algorithms exhibit varying degrees of decline compared to the performances of the TD3+BC version. Compared with the TD3+BC version, the performance of the original algorithm shows different degrees of degradation, which demonstrates that TD3+BC is more suitable for offline policy optimization than BRAC and SAC.

Environment	Task Set	FOCAL (BRAC)	CORRO (SAC)	BOReL (SAC)
Point-Robot	Train	-15.78 ± 2.89	-31.12 ± 5.82	-14.87 ± 3.68
Ant-Dir		76.47 ± 445.00	-96.06 ± 25.58	67.99 ± 42.04
Cheetah-Vel		-317.06 ± 83.96	-530.03 ± 15.43	-268.07 ± 20.59
Cheetah-Dir		722.15 ± 507.55	-50.01 ± 72.47	770.82 ± 27.19
Hopper-Params		291.60 ± 134.97	27.84 ± 0.19	184.73 ± 43.38
Walker-Params		552.04 ± 44.08	-3.68 ± 0.10	219.12 ± 44.30
Point-Robot	Test	-15.50 ± 1.02	-32.38 ± 2.14	-15.71 ± 3.27
Ant-Dir		177.45 ± 438.59	-115.99 ± 28.93	80.92 ± 27.55
Cheetah-Vel		-457.89 ± 52.19	-699.76 ± 29.79	-455.14 ± 27.78
Hopper-Params		183.86 ± 61.96	29.62 ± 0.12	152.30 ± 11.82
Walker-Params		362.68 ± 35.22	-3.26 ± 0.06	179.68 ± 29.51

Environment	Task Set	FOCAL (BRAC)	CORRO (SAC)	BOReL (SAC)
Point-Robot	Train	-17.73 ± 4.29	-29.87 ± 4.17	-21.63 ± 7.33
Ant-Dir		-42.20 ± 235.65	-103.71 ± 4.83	90.87 ± 13.03
Cheetah-Vel		-337.36 ± 22.46	-533.23 ± 8.19	-278.85 ± 34.96
Cheetah-Dir		94.04 ± 67.39	-70.38 ± 24.07	763.40 ± 12.47
Hopper-Params		203.65 ± 52.86	27.78 ± 0.09	35.70 ± 12.36
Walker-Params		378.72 ± 70.30	-3.71 ± 0.15	220.78 ± 47.08
Point-Robot	Test	-20.72 ± 2.46	-27.87 ± 5.66	-20.78 ± 1.81
Ant-Dir		-118.96 ± 324.42	-99.23 ± 11.92	64.48 ± 38.07
Cheetah-Vel		-474.81 ± 41.37	-704.84 ± 3.71	-465.04 ± 41.73
Hopper-Params		169.74 ± 51.08	29.69 ± 0.14	39.05 ± 4.24
Walker-Params		273.03 ± 49.30	-3.29 ± 0.06	170.79 ± 19.31

Table S3: Performance of the original baseline algorithms, in which FOCAL uses BRAC, and CORRO and BOReL use SAC as their backbone offline RL algorithms. Each number represents the return of the last checkpoint of the meta-policy, averaged over 3 random seeds, and $\pm$ represents standard deviation. Top: given-context performance. Bottom: one-shot performance.

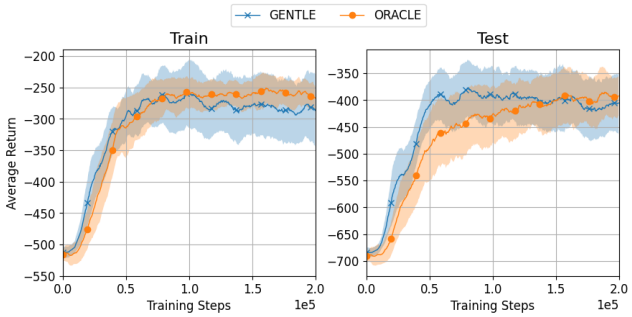

Figure S2: Return curves on training and testing tasks in Cheetah-Vel on one-shot protocol. ORACLE represents GENTLE with oracle model relabeling.

B.3 Hyperparameter Settings

To perform data relabeling for our algorithm, we train dynamic models $\{\widehat{M}_i\}_{i=1}^N$ on each offline dataset $\{D_i\}_{i=1}^N$ separately. We represent the model as a probabilistic neural network, which consists of 4 fully connected layers with 256 hidden units (expect for Point-Robot which consists of 3 forward layers with 64 hidden units). We train 7 such networks with the same architecture but different initializations to form the ensemble dynamic models. The learning rate of the models is set to 0.001. We train each ensemble dynamic model $\widehat{M}_i$ using the maximum log-likelihood estimation based on the corresponding offline dataset D_i:

$$\max_{\widehat{M}_i} \mathbb{E}_{(s,a,s',r)\sim D_i}[\log \widehat{M}_i(s', r|s, a)]. \tag{8}$$

We use a holdout ratio 0.2 to divide the training set and the validation set, and stop training when the validation error no longer decreases for 5 epochs.

Specifically, for environments where tasks differ in reward functions (Point-Robot, Cheetah-Dir, Cheetah-Vel, and Ant-Dir), we construct the model as $\widehat{M}_i(r|s, a)$ and only use (s, a, r) triplet to form the training dataset. For environments where tasks differ in dynamics and reward functions (Hopper-Params, and Walker-Params), we construct the model as $\widehat{M}_i(s', r|s, a)$ and use four-arity tuple (s, a, r, s') to form the training dataset. The dataset is also used to train TAE subsequently.

In GENTLE, TAE applies an encoder-decoder architecture to extract the representation of the tasks. The encoder and decoder are both represented by MLPs, which consist of 3 hidden layers with 256 hidden units.

The details of important configurations and hyperparameters used to produce the experimental results in the paper are presented in Table S2.

B.4 Time Complexity Analysis

To investigate the time complexity of the algorithms, we run GENTLE, FOCAL, and CORRO on the same dataset and machine, and results are shown in Table S4. GENTLE's pretraining (0.29h) and relabeling (0.02h) incur little time overhead, and its training time is slightly longer than that of FOCAL.

Run Time (hours)	GENTLE	FOCAL	CORRO
Pretrain	0.29	0.00	0.60
Train	1.59 (0.02)	1.47	0.51
Evaluate	1.93	1.99	2.13

Table S4: Run time of GENTLE, FOCAL and CORRO.

B.5 Additional Experiment Results

Assuming that we have access to the transition function and the reward function of each task, which we term as the oracle model. We can leverage this oracle model for data relabeling. To explain further, we replace the dynamics model trained via supervised learning with the oracle model for dynamics-relabeling. We carry out the experiments on Cheetah-Vel, and the results are depicted in Figure S2. It indicates that the accuracy of relabeling data has a certain influence on the performance. Utilizing a precise oracle model for relabeling can lead to more stable performance.