A Prompt-based Contrastive Learning

To construct domain-invariant representations efficiently for egocentric perception data in embodied agent environments, we adopt contrastive tasks for visual prompt learning on the domain factors, which can be learned from a few expert demonstrations.

When conducting prompt-based contrastive learning, as shown in Figure 1 and explained below, we specifically adopt several methods to generate positive visual observation pairs for different domain factors.

Consider a pretrained model $\mathcal{T}_\phi$ parameterized by ϕ that maps observations $o \in \Omega$ to the embedding space $\mathcal{Z}$. The contrast function $P : \Omega \times \Omega \to \{0, 1\}$ discriminates whether an observation pair is positive or not. Then, we fine-tune $\mathcal{T}_\phi$ by learning a visual prompt p^v through contrastive learning, where the contrastive loss function [1] is defined as Equation (2) in the main manuscript.

Behavior-driven Contrast. Similar to [2], we exploit expert actions to obtain positive sample pairs from expert trajectories of different domains. With observation and action pairs $(o, a), (o', a')$, a behavior-driven contrast function is defined as $P_{\text{beh}}(o, o') = \mathbb{1}_{a=a'}$. If the environment has a discrete action space, the behavior-driven contrast can be applied immediately to obtain positive sample pairs; otherwise, it can be applied after discretizing continuous actions with unsupervised clustering algorithms such as k-means clustering [3].

Augmentation-driven Contrast. Similar to visual domain randomization techniques [4, 5], we use data augmentation (e.g., color perturbation [6]) for unstructured pixel-level visual domain factors such as illumination. An augmentation-driven contrast function is defined as $P_{\text{aug}}(o, o') = \mathbb{1}_{o'=AUG(o)}$, where AUG augments o.

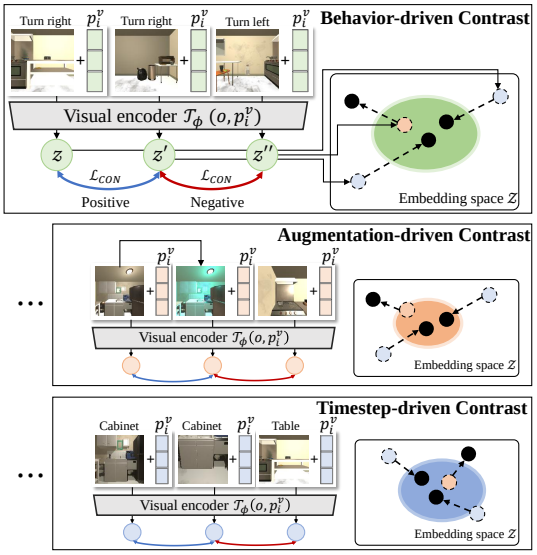


Figure 1: Prompt-based Contrastive Learning with Different Contrastive Tasks

Timestep-driven Contrast. Similar to [7], we exploit timesteps of expert trajectory to obtain positive sample pairs across different domains. With observation and timestep pairs $(o, t), (o', t')$, a timestep-driven contrast function is defined as $P_{\text{tim}}(o, o') = \mathbb{1}_{t=t'}$, where $t - k \leqq t' \leqq t + k$ and k is hyperparameter.

B Prompt Ensemble with a Pretrained Policy

As mentioned in the main manuscript, we devise an optimization method to update $\mathcal{G}$ specifically for a pretrained policy π. Specifically, we use a *policy prompt* p^v_{pol} that focuses on task-relevant features from observations for π. By incorporating the prompted embedding $\tilde{z}_0$, which contains these task-relevant features, into the guided-attention-based ensemble, we can effectively integrate the policy π with the attention module, resulting in $\pi(\mathcal{G}(\tilde{z}_0, \mathbf{z}))$. Here, $\tilde{z}_0$ is obtained by applying the transformation to the observation o using the policy prompt p^v_{pol}.

As such, CONPE enables efficient adaptation of the attention module to a pretrained policy by fine-tuning only a small number of parameters. This achieves robust zero-shot performance for unseen domains in different tasks.

Algorithm 1 shows the procedures of CONPE to adapt the attention module for a pretrained policy. The first half corresponds to prompt-based contrastive learning and the other half corresponds to learning of the attention module $\mathcal{G}$ with a pretrained policy $\pi(Z)$. This is slightly extended from the algorithm in the main manuscript, where joint learning of the attention module and a policy is explained.

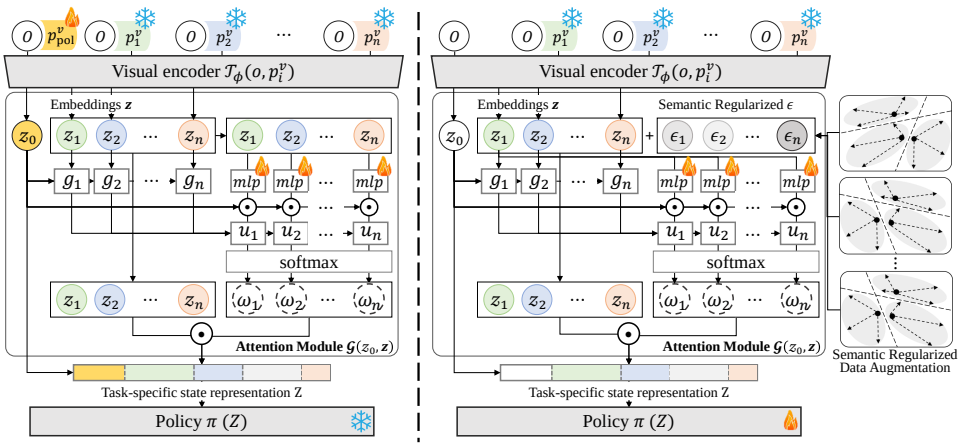


Figure 2: Guided-Attention-based Prompt Ensemble. The cosine similarity-guided attention module $\mathcal{G}$ generates task-specific state representations by combining multiple prompted embeddings, and it is learned with a policy network π. The left part of the figure illustrates prompt ensemble adaptation to a pretrained policy, while the right part shows the semantic regularized data augmentation scheme.

Algorithm 1 Procedure in CONPE with a Pretrained Policy

Dataset $\mathcal{D} = \{(o_1, o_1'), ...\}$, replay buffer $Z_D \leftarrow \emptyset$, pretrained vision-language model $\mathcal{T}_\phi$
Visual prompt pool $\mathbf{p}^v = [p_1^v, ..., p_n^v]$, attention module $\mathcal{G}$
Pretrained policy π

/ Prompt-based Contrastive Learning */*
for $i = 1, ..., n$ **do**
  while not converge **do**
    Sample a batch $\mathcal{B}_{P_i} = \{(o_j, o_j')\}_{j \leq m} \sim \mathcal{D}$
    Update prompt $p_i^v \leftarrow p_i^v - \nabla \mathcal{L}_{\text{CON}}(p_i^v, \mathcal{B}_{P_i})$ using main manuscript Equation (3)
  end while
end for
/ Prompt Ensemble Learning with a Pretrained Policy */*
for each environment step **do**
  Sample action $a = \pi(\mathcal{G}(\mathcal{T}_\phi(o), \mathbf{z}))$ using main manuscript Equation (5), (6)
  Store $Z_D \leftarrow Z_D \cup \{(\mathbf{z}, a, r)\}$
  Optimize module $\mathcal{G}$ on $\{(\mathbf{z}_j, a_j, r_j)\}_{j \leq m} \sim Z_D$
end for

C Semantic Regularized Data Augmentation

For a source environment that is sufficiently accessible, the attention module and policy network can be jointly trained by RL algorithms. In this case, to address overfitting problems to the source environment, data augmentation methods can be adopted. For example, when training a policy, it is feasible to add Gaussian noise to each prompted embedding as part of data augmentation to avoid overfitting [8, 9].

To enhance both policy optimization and zero-shot performance, we investigate semantic regularization schemes in the CLIP embedding space, which are specific to the prompt ensemble-based policy learning. Specifically, using a few object-level descriptions in datasets, we control the noise effectively.

In our semantic regularization, the language prompt $p_i^l = [e_1^l, e_2^l, ..., e_{u'}^l]$, $e_i^l \in \mathbb{R}^{d'}$ is pretrained with description data and fixed p_i^v. Then, p_i^l is used as a semantic regularizer, where e_i^l is a continuous learnable vector of the word embedding dimension d' (e.g., 512 for CLIP language encoder) and u' is the length of a language prompt. Similar to [10], we adopt language prompt learning schemes. We

also use the binary cross entropy loss for observations and object-level description pairs (o, m),

$$\mathcal{L}_{\text{BCE}}(p_i^v, p_i^l) = \sum_{k=1}^{n} \sum_{q \in \Phi} [\log f(\mathcal{T}_\phi(o_k, p_i^v), \mathcal{T}_\phi(q, p_i^l)) - \mathbb{1}_{q \in m_k} \log f(\mathcal{T}_\phi(o_k, p_i^v), \mathcal{T}_\phi(q, p_i^l))] \quad (1)$$

where Φ is the collection of object-level descriptions and f is a cosine similarity function.

Given representation $z_i = \mathcal{T}_\phi(o, p_i^v)$, we add a small Gaussian noise $\epsilon \sim \mathcal{N}(0, \delta)$ with variance δ to z_i. For all object-level descriptions q_i in the given Φ, we maintain regularizing augmented representations to hold the below condition.

$$\text{CL}(z_i + \epsilon, \mathcal{T}_\phi(q_i, p_i^l)) = \text{CL}(z_i, \mathcal{T}_\phi(q_i, p_i^l)), \text{ where } \text{CL}(z, z') = \mathbb{1}_{\{\sigma(S(z,z')) \geq 0.5\}}. \quad (2)$$

This tends to achieve more generalized representations for a specific domain that is relevant to the object-level descriptions, while maintaining semantic information in the representations.

Algorithm 2 shows the procedure of CONPE with semantic regularized data augmentation. This algorithm includes three steps: the first step corresponds to prompt-based contrastive learning (same as the algorithm in the main manuscript), the second step corresponds to language prompt learning (addition for this algorithm), and the third step corresponds to the modified process of joint learning for policy $\pi(Z)$ and the attention module $\mathcal{G}$ with semantic regularized data augmentation.

Algorithm 2 Procedure of CONPE with Semantic Regularized Data Augmentation

Dataset $\mathcal{D} = \{(o_1, o_1', m), ...\}$, replay buffer $Z_D \leftarrow \emptyset$, pretrained vision-language model $\mathcal{T}_\phi$
Visual prompt pool $\mathbf{p}^v = [p_1^v, ..., p_n^v]$, Language prompts $p_1^l, ..., p_n^l$
Attention module $\mathcal{G}$, policy π

/ Prompt-based Contrastive Learning */*
for $i = 1, ..., n$ **do**
  while not converge **do**
    Sample a batch $\mathcal{B}_{P_i} = \{(o_j, o_j')\}_{j \leq m} \sim \mathcal{D}$
    Update prompt $p_i^v \leftarrow p_i^v - \nabla \mathcal{L}_{\text{CON}}(p_i^v, \mathcal{B}_{P_i})$ using main manuscript Equation (3)
  end while
end for
/ Language Prompt Learning */*
for $i = 1, ..., n$ **do**
  while not converge **do**
    Sample a batch $\{(o_k, m_k)\}_{k \leq B} \sim \mathcal{D}$
    Update prompt $p_i^l \leftarrow p_i^l - \nabla \mathcal{L}_{\text{BCE}}(p_i^v, p_i^l)$ using (1)
  end while
end for
/ Prompt Ensemble-based Policy Learning with Semantic Regularized Data Augmentation*/*
for each environment step **do**
  Sample $\boldsymbol{\epsilon} = [\epsilon_1, ..., \epsilon_n]$ satisfying the condition (2)
  Compute $\mathbf{z}_\epsilon = \mathbf{z} + \boldsymbol{\epsilon} = [z_1 + \epsilon_1, ..., z_n + \epsilon_n]$
  Sample action $a = \pi(\mathcal{G}(\mathcal{T}_\phi(o), \mathbf{z}))$ using main manuscript Equation (5), (6)
  Store $Z_D \leftarrow Z_D \cup \{(\mathbf{z}, a, r)\}$
  Jointly optimize policy π and module $\mathcal{G}$ on $\{(\mathbf{z}_j, a_j, r_j)\}_{j \leq m} \sim Z_D$
end for

D Environments and Datasets

D.1 AI2THOR

Environment settings. We use AI2THOR [11], a large-scale interactive simulation platform for Embodied AI. In AI2THOR, we use iTHOR datasets that have 120 room-sized scenes with bedrooms, bathrooms, kitchens, and living rooms. iTHOR includes over 2000 unique objects based on Unity 3D. Among embodied AI tasks in AI2THOR, we evaluate our framework with object goal navigation and point goal navigation tasks. We also test the image goal navigation task, a modified version of the object goal navigation, as well as the room rearrangement task for adaptation to a pretrained policy.

Table 1: AI2THOR Actions

Actions	
Navigation	Move [Ahead/Left/Right/Back] Rotate [Right/Left] Look [Up/Down] Done
Object Interaction	PickUp [Object Type] Open [Object Type] PlaceObject

Object Goal Navigation. The object navigation task requires an agent to navigate through its environment and find an object of a given category (e.g., apple). The agent is initially placed at a random location in a near-photorealistic home, and it receives an egocentric viewpoint image and one target object instruction for each timestep. The agent uses several navigation actions such as MoveAhead and RotateRight to complete the task.

Point Goal Navigation. In the point goal navigation task, an agent is given a specific coordinate in the environment as its target goal. Similar to the object goal navigation task, the agent receives an egocentric viewpoint image and a target coordinate for each timestep.

Image Goal Navigation. In the image goal navigation task, an agent is provided with a target image that represents a desired scene configuration. The agent's goal is to navigate through the environment and reach a location where the observed scene matches the target image. This task involves using visual perception to compare the current scene to the target image and selects appropriate navigation actions to achieve the desired scene configuration. The agent receives egocentric viewpoint images and target image for each timestep. The task is considered successful when the agent reaches a specific position where the observed scene closely resembles the target image.

Room Rearrangement. In the room Rearrangement task, the goal of an agent is to reach the goal configuration by interacting with the environment. At each timestep, the images of both the current state and goal state are given, and the agent uses navigation actions and higher-level semantic actions (e.g., PickUp CUP) to rearrange the objects and recover the goal room configuration.

Our experiment settings. We adopt the similar configuration in [12] for our experiments, while some settings are modified to evaluate zero-shot adaptation for different domains.

We use "FloorPlan21" as our default environment. For zero-shot adaptation scenarios, 75 different domains are randomly generated with several predefined domain factors (i.e., camera field of views, stride length, rotation degrees, look degrees and illuminations). These factors, previously examined in embodied RL studies [13, 14, 15], can be characterized by either discrete or continuous values based on their intrinsic properties. For instance, we treat rotation degrees as a discrete factor, determined based on the feasibility of task success; conversely, brightness is treated as a continuous factor, with a range extending from 0.0 to 1.0. Each of these domain factors is randomly selected from a uniform distribution, leading to a combination of varied domains. Using the domain factors, we define several seen domains that can be used for representation and policy learning as well as unseen domains for evaluating the zero-shot adaptation performance.

Datasets. Using rule-based policies, we create expert datasets for contrastive representation learning. The datasets comprise 28,464 samples for AI2THOR. Table 11 illustrates the examples of the expert datasets in which the experts of each domain reflect external differences amongst domains and physical differences of agents. SPL is the evaluation metric, success weighted by (normalized inverse) path length [16]. LENGTH is the average episode length of entire trajectories.

D.2 Egocentric-Metaworld

Environment settings. The Metaworld benchmark [17] includes diverse table-top manipulation tasks that require a Sawyer robot to interact with various objects. With different objects, such as door and button, the robot needs to manipulate them based on the object's affordance, leading to different reward functions. At each timestep, the Sawyer robot conducts a 4-dimensional fine-grained action that determines the 3D position movements of the end-effector and the variation of gripper openness.

For embodied AI settings, we slightly modify Metaworld to have egocentric images as observations. We use several tasks such as reach-v2, reach-wall-v2, button-press-topdown-v2 and door-open-v2 for our experiments.

Reach. In the Reach task, the objective is to control a Sawyer robot's end-effector to reach a target position. The agent directly controls the XYZ location of the end-effector.

Reach-Wall. In the Reach-Wall task, the agent controls the Sawyer robot's end-effector to reach a target position in the presence of obstacles such as walls. The agent needs to plan and navigate a path that avoids collisions to the walls while reaching the target.

Button-Press. In the Button-Press task, the agent is required to accurately guide the Sawyer robot's end-effector to a designated button and press it. This task involves precise control and coordination to successfully interact with the button.

Door-Open. In the Door-Open task, the agent's objective is to manipulate a Sawyer robot's end-effector to open a door. The agent needs to grasp and manipulate the door handle to open the door.

Our experiment settings. For zero-shot adaptation scenarios, 70 different domains are randomly generated with predefined domain factors such as camera positions, gravity, wind speeds, and illuminations. Each domain factor can be represented as either discrete or continuous values, depending on its inherent nature. Regarding sampling methods, these domain factors are individually drawn from a uniform distribution to produce combinatorial domain variations.

Datasets. For predefined seen domains, we implement a rule-based expert policy to collect expert trajectory data. The datasets comprise 3,840 samples for Metaworld. Table 12 illustrates a few examples of our expert dataset where experts of each domain reflect external differences among domains and physical differences in agents.

D.3 CARLA

Environment settings. CARLA [18] is a self-driving simulation environment where an agent navigates to the target location while avoiding collisions and lane crossings. For experiments, we use the CARLA simulator v0.9.13 and choose Town10HD as our map. For RL formulation, we incorporate the RGB image data and sensor values (acceleration, velocity, angular velocity) into states, and use control steer, throttle, and brake as actions. Each action ranges from -1 to 1. The actions are automatically calibrated when the speed of the car reaches the maximum. The agent is evaluated based on the reward function below that involves the desired velocity and goal distance,

$$\mathbf{reward} = v_t \cdot \frac{v_{target}}{\|v_{target}\|} - \frac{goal\_distance}{100} \tag{3}$$

where v_t denotes the agent's current velocity and v_{target} denotes the target velocity.

Table 2: CARLA Environment Configuration

Configures	Value
Observation space Ω	$[0,1]^{224\times224\times3} \times [-1,1]^9$
Action space A	$[-1,1]^2$
Maximum speed	20km/h

Our experiment settings. To implement 50 different domains, we use camera positions, camera field of views, weather conditions, times of day, and different ranges of action magnitude as domain factors. Each domain factor can be represented as either discrete or continuous values, depending on its inherent nature. For example, we treat weather conditions as a discrete factor, which can be classified as either clear, rainy, cloudy, or other. Through these factors, we define several seen domains that can be used for representation and policy learning as well as unseen domains for evaluating the zero-shot adaptation performance. For prompt-based contrastive learning, the training dataset consists of a single trajectory for each of 50 domains. In policy learning, we utilize 4 source domains across 2 different maps.

Datasets. For predefined seen domains, we implement a rule-based expert policy to collect expert trajectory data. The datasets comprise 7,394 samples for CARLA. The detailed information of the expert dataset is explained in Table 13.

(a) Source Domain (b) Seen Target Domain (c) Unseen Target Domain

Figure 3: Examples from the Source, Seen Target, and Unseen Target Domains. (a) represents the source domain, (b) depicts the seen target domain with an altered camera position, and (c) showcases the unseen target domain, differing from both the source and seen target domains.

E Implementation Details

In this section, we present the implementation details of our proposed framework CONPE and each baseline method. CONPE is implemented using Python v3.7, Jax v0.3.4, and PyTorch v1.13.1, and is trained on a system of an Intel(R) Core (TM) i9-10980XE processor and an NVIDIA RTX A6000 GPU.

E.1 LUSR

LUSR is a domain adaptation method that utilizes the latent embedding of encoder-decoder models to extract generalized representations. LUSR uses β-VAE to learn disentangled representations of different visual domains. For implementation, we use the open source (https://github.com/KarlXing/LUSR). When implementing LUSR, we use a CNN encoder for both DAE and β-VAE. We conduct online policy learning with PPO algorithms [19]. The hyperparameter settings are summarized in Table 3.

Table 3: Hyperparameter Settings for LUSR

(a) Hyperparameters for Representation Learning

Hyperparameters	**Value**
image size	(3, 224, 224) RGB
batch size	10
train epochs	200
optimizer	Adam
learning rate	$1e-4$
beta β	10
latent size	32

(b) Hyperparameters for Policy Learning

Hyperparameters	**Value**
observation	(3, 224, 224) RGB
discount factor	0.99
GAE parameter	0.95
clipping parameter	0.1
value loss coefficient	0.5
entropy loss coefficient	0.01
learning rate	$3e-4$
optimizer	Adam
training steps	3M
steps per rollout	500

E.2 CURL and ATC

CURL and ATC are a contrastive learning based framework for visual RL. CURL uses contrastive representation learning to extract discriminative features from raw pixels which greatly

enhance sample efficiency in RL training. ATC enables the training of an encoder to associate pairs of observations separated by short time difference, leading to RL performance enhancement. For implementation, we use the open source (https://github.com/facebookresearch/moco) and (https://github.com/astooke/rlpyt/tree/master/rlpyt/ul). The hyperparameter settings of CURL and ATC are summarized in Table 4 and 5, respectively, where the other settings are the same as in Table 3(b).

Table 4: Hyperparameter Settings for CURL

Hyperparameters	**Value**
image size	(3, 224, 224) RGB
batch size	256
model architecture	ViT-B/32
train epochs	500
optimizer	sgd
learning rate	$1e-3$

Table 5: Hyperparameter Settings for ATC

Hyperparameters	**Value**
image size	(3, 224, 224) RGB
batch size	256
model architecture	ViT-B/32
train epochs	500
optimizer	sgd
timestep k	3
learning rate	$1e-3$

E.3 ACO

ACO utilizes augmentation-driven and behavior-driven contrastive tasks in the context of RL. For implementation, we use the open source (https://github.com/metadriverse/ACO). The hyperparameter settings are summarized in Table 6, where other settings are the same as in Table 3(b).

Table 6: Hyperparameter Settings for ACO

Hyperparameters	**Value**
image size	(3, 224, 224) RGB
batch size	256
model architecture	ViT-B/32
train epochs	500
optimizer	sgd
learning rate	$1e-3$

E.4 EmbCLIP

EmbCLIP is a state-of-the-art model for embodied AI tasks. By using CLIP as the visual encoder, EmbCLIP extracts generalized representation which is useful for an embodied agent, enabling the agent to effectively generalize to different environments and tasks. We use the open source (https://github.com/allenai/embodied-clip) for the implementation of AI2-THOR environments. To evaluate this with the CARLA simulator, we also use the open source (https://github.com/openai/CLIP). The configurations for policy learning are the same as in Table 3(b).

E.5 ConPE

The procedure of our CONPE consists of prompt learning and policy learning steps.

Prompt-based Contrastive Learning. In prompt learning step, for sample-efficiency, CONPE conducts prompt-based contrastive learning, exploiting the pretrained CLIP model. We set the length of visual prompts to be 8 for each contrastive learning with Equation (2) in the main manuscript. In cases when metadata is available (the cases of using the semantic regularized data augmentation), 8 language prompts are used for (1). The hyperparameter settings are summarized in Table 7.

Prompt Ensemble-based Policy Learning. CONPE obtains domain-invariant states from observations using the ensemble of multiple prompts. The prompt attention module $\mathcal{G}$ consists of as many

Table 7: Hyperparameter Settings for CONPE's Prompt-based Contrastive Learning

(a) Augmentation-driven

Hyperparameters	Value
image size	(3, 224, 224) RGB
batch size	256
visual prompt length	8
pretrained model	CLIP ViT-B/32
train epoch	1000
optimizer	Adam
learning rate	$1e-2$

(b) Behavior-driven

Hyperparameters	Value
image size	(3, 224, 224) RGB
batch size	64
visual prompt length	8
pretrained model	CLIP ViT-B/32
train epoch	500
optimizer	Adam
learning rate	$1e-2$

(c) Timestep-driven

Hyperparameters	Value
image size	(3, 224, 224) RGB
batch size	64
visual prompt length	8
pretrained model	CLIP ViT-B/32
train epoch	500
optimizer	Adam
timestep k	3
learning rate	$1e-2$

MLP(Multi-Layer Perceptron) layers as the number of the prompts and it is learned jointly with a policy network for a given task. The hyperparameter settings for policy learning are the same as in Table 3(b).

F Additional Experiments

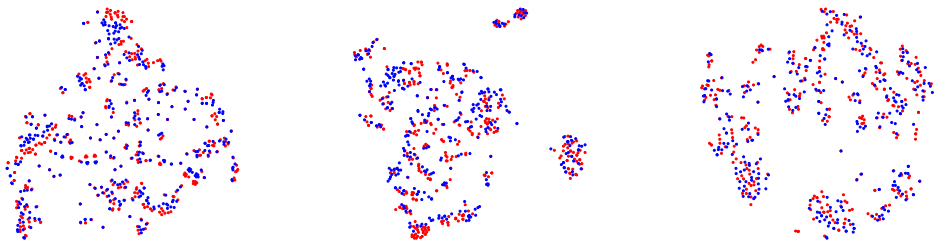
F.1 Zero-shot Performance for Seen Domains Factors

Table 8 presents the zero-shot performance of CONPE across specific domain factors (DF). For example, DF0 refers to various domains where the camera position is a domain factor of interest, and LUSR's 48.5 in DF0 indicates the success rate of LUSR for the domains generated by different camera positions.

As shown, CONPE maintains robust zero-shot performance for all the cases (DF0$\sim$ DF9), compared to the baselines. Additionally, as shown in Figure 5 where a specific domain factor is changed, the attention weights assigned to the prompted embedding (denoted as Pn, where $n = \{0..9\}$) that is trained for the corresponding domain factor are high (in the bright color).

Table 8: Zero-shot Performance for Seen Domains Factors

Method	DF0	DF1	DF2	DF3	DF4	DF5	DF6	DF7	DF8	DF9
LUSR	48.5	37.2	27.5	66.4	69.5	41.6	11.7	40.6	26.8	48.1
CURL	28.6	27.3	8.4	28.3	13.0	14.9	2.2	13.2	11.8	26.6
ATC	73.2	84.9	73.1	89.8	95.4	79.8	55.8	89.8	69.0	88.1
ACO	38.3	39.6	36.0	37.2	35.9	30.8	21.5	31.1	24.1	41.1
EmbCLIP	71.6	79.3	83.5	92.3	96.8	**90.8**	62.0	92.7	**75.8**	92.4
CONPE	**78.1**	**90.9**	**93.2**	**95.5**	**97.0**	88.2	**68.7**	**93.9**	67.0	**93.7**



(a) Domain Factor 1 (b) Domain Factor 2 (c) Domain Factor 3

Figure 4: Intra Prompted Embeddings. Two distinct domains, represented in blue and red dots in the figures, are chosen based on different configurations of the same domain factor to assess their embedding alignment.

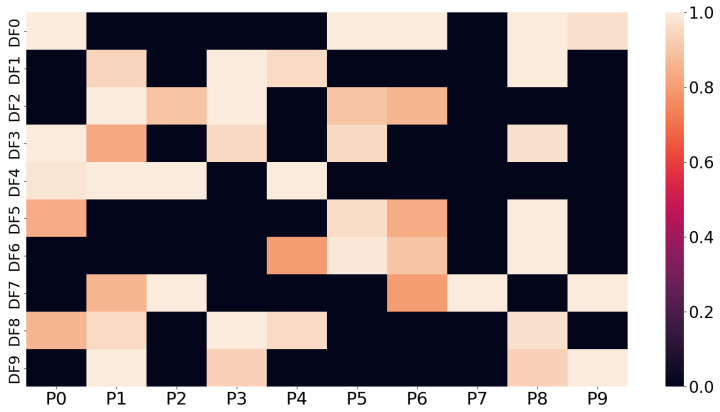


Figure 5: Prompt Ensemble Attention Weight Matrix for Seen Domains Factors.

F.2 Prompt Ensemble with a Pretrained Policy

Table 9 reports detailed zero-shot performance for the scenarios when a pretrained policy is given. We additionally test widely used baselines that leverage large pretrained models in the fields of vision and natural language, specifically in the context of prompt-meta learning. ATTEMPT [20] is a parameter-efficient multi-task language model tuning method that transfers knowledge across different tasks via a mixture of soft prompts. SESoM [21] is a soft prompts ensemble method that leverages multiple source tasks and effectively improves few-shot performance of prompt tuning by transferring knowledge from the source tasks to a target task. CONPE demonstrates the ability to effectively improve zero-shot performance with a small number of samples, especially when a pretrained policy is given. As shown in Figure 7 (a) and (b), prompt ensemble adaptation demonstrates an increase in success rate with a small number of samples in both the train and unseen environments of the pretrained policy, compared to other baselines. This results present the sample-efficiency of our prompt ensemble adaptation method.

Table 9: Prompt Ensemble with a Pretrained Policy.

(a) Zero-shot Performance in AI2THOR with Visual Navigation and Room Rearrangement Tasks

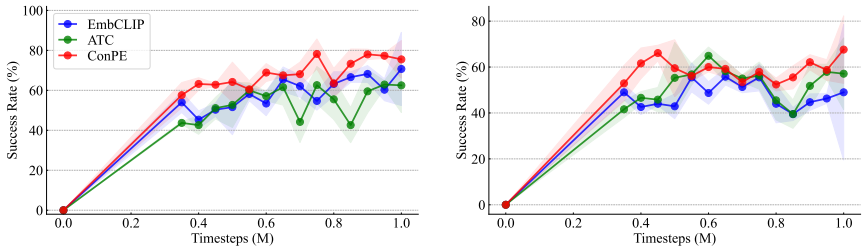
Method	ObjectNav. (Aln.)		PointNav. (Not Aln.)		ImageNav. (Not Aln.)		RoomR. (Not Aln.)	
	Source	Target	Source	Target	Souce	Target	Scoure	Target
Pretrained	87.5±17.2	65.8±19.1	95.3±4.6	80.9±1.6	77.2±3.3	56.2±2.2	87.3±3.1	75.2±13.2
ATTEMPT	2.85±0.4	3.24±0.3	20.2±0.5	20.7±0.3	13.8±3.0	15.0±2.3	5.3±1.2	3.6±1.6
SESoM	2.0±6.6	3.4±5.0	19.7±0.5	20.6±0.1	11.2±2.4	8.9±1.2	60.0±2.0	44.2±14.0
CONPE	88.4±1.7	**72.8±3.1**	98.9±1.0	**84.4±1.0**	79.2±1.4	**61.6±1.1**	93.3±1.2	**82.2±14.4**

(b) Zero-shot Performance in Egocentric-Metaworld with 4 Different Robot Manipulation Tasks

Method	Reach (Aln.)		Reach-Wall (Not Aln.)		Button-Press (Not Aln.)		Door-Open (Not Aln.)	
	Source	Target	Source	Target	Source	Target	Source	Target
Pretrained	100.0±0.0	65.7±6.4	100.0±0.0	58.0±5.8	100.0±0.0	16.8±2.3	100.0±0.0	35.6±6.2
ATTEMPT	73.3±5.8	33.7±7.7	76.7±15.3	38.0±4.0	100.0±0.0	25.7±8.3	100.0±0.0	44.0±7.4
SESoM	16.7±5.8	9.3±2.1	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
CONPE	100.0±0.0	**74.7±5.0**	100.0±0.0	**75.7±9.0**	100.0±0.0	**73.7±8.3**	100.0±0.0	**93.2±1.1**

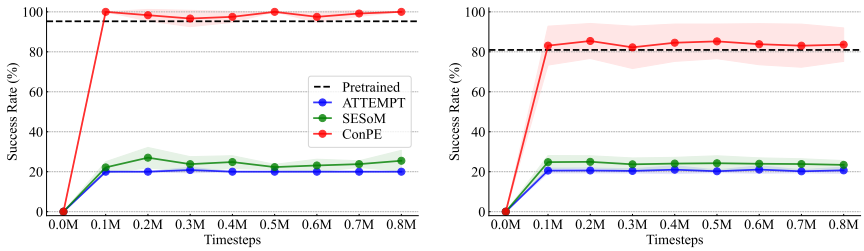
F.3 Semantic Regularized Data Augmentation

Table 10 shows the zero-shot performance for semantic regularized data augmentation in CONPE, where language data is additionally used. As shown, CONPE with language data (w Semantic Reg.) consistently yields better performance over CONPE without language data (w/o Semantic Reg.) across various noise scales (δ). As the deviation (δ) of the Gaussian noise varies, it is observed that larger deviation does not necessarily lead to performance improvement. This indicates that



(a) Success Rate of Train Environment (b) Success Rate of Unseen Environment

Figure 6: Sample-efficiency of Prompt Ensemble-based Policy Learning (up to 1 million timesteps). These detailed evaluation graphs focused on the initial part of the training, are consistent with the experiment in Figure 4 of the manuscript.



(a) Success Rate of Train Environment (b) Success Rate of Unseen Environment

Figure 7: Sample-efficiency of Prompt Ensemble with a Pretrained Policy. The pretrained policy is learned on the Object Goal Navigation task and then adapted to the Point Goal Navigation task with *policy prompt*.

enhancing data augmentation diversity through higher noise deviation may not always be beneficial. However, when maintaining the semantics (w Semantic Reg.), the performance can improve with larger deviation.

Table 10: Semantic Regularized Data Augmentation.

δ	w/o Semantic Reg.			w Semantic Reg.		
	Source	Seen Target	Unseen Target	Source	Seen Target	Unseen Target
0	90.8±10.9%	72.1±6.5%	80.0±7.2%	90.8±10.9%	78.0±9.6%	80.0±7.2%
0.1	97.4±3.8%	84.5±8.3%	82.7±9.4%	**100.0±0.0%**	**86.0±6.2%**	**82.1±14.2%**
0.2	94.7±0.0%	82.1±9.5%	73.1±16.1%	**94.8±7.4%**	**84.2±6.2%**	**75.1±11.9%**
0.3	84.2±3.7%	77.4±6.6%	73.1±10.9%	**96.1±1.9%**	**86.1±4.7%**	**80.1±15.2%**
0.4	80.3±16.2%	75.8±12.1%	72.2±17.7%	**86.9±3.8%**	**80.5±8.6%**	**76.0±15.8%**
0.5	71.1±9.6%	64.5±12.8%	59.1±14.2%	**73.7±3.8%**	**68.4±4.6%**	**66.7±9.8%**

References

[1] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[2] Minbeom Kim et al. "Action-driven contrastive representation for reinforcement learning". In: *PLOS ONE* (2022).

[3] John A Hartigan and Manchek A Wong. "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the royal statistical society. series c (applied statistics)* (1979).

[4] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 1597–1607.

[5] Nicklas Hansen and Xiaolong Wang. "Generalization in reinforcement learning by soft data augmentation". In: *Proceedings of the 38th International Conference on Robotics and Automation*. IEEE. 2021, pp. 13611–13617.

[6] Adrian Galdran et al. "Data-driven color augmentation techniques for deep skin image analysis". In: *arXiv preprint arXiv:1703.03702* (2017).

[7] Adam Stooke et al. "Decoupling representation learning from reinforcement learning". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021, pp. 9870–9879.

[8] Denis Yarats, Ilya Kostrikov, and Rob Fergus. "Image Augmentation Is All You Need: Regularizing deep reinforcement learning from pixels". In: *Proceedings of the 9th International Conference on Learning Representations*. 2021.

[9] Samarth Sinha, Ajay Mandlekar, and Animesh Garg. "S4RL: Surprisingly Simple Self-Supervision for Offline Reinforcement Learning in Robotics". In: *Proceedings of the 5th Conference on Robotics Learning*. 2021, pp. 907–917.

[10] Kaiyang Zhou et al. "Learning to prompt for vision-language models". In: *International Journal of Computer Vision* (2022).

[11] Eric Kolve et al. "Ai2-thor: An interactive 3d environment for visual ai". In: *arXiv preprint arXiv:1712.05474* (2017).

[12] Apoorv Khandelwal et al. "Simple but Effective: CLIP embeddings for embodied AI". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14809–14818.

[13] Prithvijit Chattopadhyay et al. "Robustnav: Towards benchmarking robustness in embodied navigation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15691–15700.

[14] Dhruv Shah et al. "Gnm: A general navigation model to drive any robot". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE. 2023, pp. 7226–7233.

[15] Ryan Julian et al. "Efficient adaptation for end-to-end vision-based robotic manipulation". In: *Proceedings of the 4th Lifelong Machine Learning Workshop at ICML*. 2020.

[16] Peter Anderson et al. "On evaluation of embodied navigation agents". In: *arXiv preprint arXiv:1807.06757* (2018).

[17] Tianhe Yu et al. "Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning". In: *Proceedings of the 3rd Conference on Robot Learning*. 2019, pp. 1094–1100.

[18] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Proceedings of the 1st Conference on Robot Learning*. 2017, pp. 1–16.

[19] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[20] Akari Asai et al. "Attempt: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2022, pp. 6655–6672.

[21] Xiangyu Peng et al. "Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning". In: *arXiv preprint arXiv:2210.12587* (2022).

Table 11: AI2THOR Expert Dataset

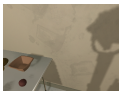
ID	Observation	Environmental Difference				Physical Property				Expert Episode	
		Brightness	Contrast	Saturation	Hue	Camera	Step Size	Rotation Degree	Look Degree	SPL	Length
1		1.0	1.0	1.0	0.0	63.5	0.25	30.0°	30.0°	0.90	9.58
2		1.0	1.0	1.0	0.0	29.7	0.25	30.0°	30.0°	0.92	23.30
3		1.0	1.0	1.0	0.4	63.5	0.25	30.0°	30.0°	0.90	9.50
4		1.0	1.0	1.0	0.0	63.5	0.25	5.0°	30.0°	0.91	24.90
5		1.0	1.0	1.0	0.0	46.0	0.25	30.0°	30.0°	0.88	35.80
6		1.0	1.0	1.7	0.0	63.5	0.25	30.0°	30.0°	0.90	9.58
7		1.0	1.0	1.0	0.0	63.5	0.01	30.0°	30.0°	0.97	110.0
8		1.0	3.3	1.0	0.0	63.5	0.25	30.0°	30.0°	0.90	9.58
9		1.0	1.0	1.0	0.0	92.9	0.25	30.0°	30.0°	0.84	8.82
10		1.0	1.0	1.0	0.0	127.0	0.25	30.0°	30.0°	0.82	8.58

Table 12: Egocentric-Metaworld Expert Dataset

ID	Observation	Environmental Difference				Physical Property			Expert Episode	
		Brightness	Contrast	Saturation	Hue	Camera Position	Wind	Gravity	Rewards	Length
1		1.0	1.0	1.0	0.0	1.0	0.0	0.0	315.48	48.00
2		1.0	1.0	1.0	0.0	2.0	0.0	0.0	315.48	48.00
3		1.0	1.0	1.0	0.0	2.0	0.0	9.0	199.14	35.00
4		1.0	1.0	1.0	0.0	3.0	0.0	1.0	315.48	48.00
5		1.0	3.3	1.0	0.0	2.0	0.0	1.0	315.48	48.00
6		1.0	1.0	1.7	0.0	2.0	0.0	1.0	315.48	48.00
7		1.0	1.0	1.0	0.0	2.0	8.0	1.0	387.49	56.00
8		1.0	1.0	1.0	0.4	2.0	0.0	1.0	315.48	48.00
9		1.5	1.0	1.0	0.0	2.0	0.0	1.0	315.48	48.00
10		1.0	1.0	1.0	0.0	2.0	0.0	1.0	252.86	41.00

Table 13: CARLA Expert Dataset

ID	Observation	Environmental Difference		Physical Property			Expert Episode	
		Weather	Daytime	Control Sensitivity	Camera Position	Field of View	Rewards	Length
1		Clear	Sunset	(100%, 100%, 100%)	high	75 °	2691.8	758
2		Clear	Noon	(100%, 85%, 100%)	low	60 °	2713.2	749
3		Cloudy	Night	(85%, 100%, 85%)	low	90 °	2747.9	733
4		Clear	Sunset	(100%, 100%,100%)	high	110 °	2742.8	733
5		Cloudy	Noon	(100%, 85%, 100%)	low	60 °	2746.6	736
6		MidRainy	Sunset	(70%, 70%, 85%)	low	60 °	2736.9	734
7		MidRainy	Noon	(100%, 85%, 70%)	low	60 °	2735.2	736
8		SoftRainy	Nigjht	(70%, 70%, 70%)	low	60 °	2716.8	746
9		Cloudy	Sunset	(70%, 70%, 100%)	low	60 °	2739.8	731
10		Clear	Sunset	(100%, 100%, 100%)	high	95 °	2725.7	738