Rethinking Latent Redundancy in Behavior Cloning: An Information Bottleneck Approach for Robot Manipulation

Shuanghao Bai [1] **Wanqi Zhou** [1] **Pengxiang Ding** [2 3] **Wei Zhao** [2] **Donglin Wang** [2] **Badong Chen** [1]

Abstract

Behavior Cloning (BC) is a widely adopted visual imitation learning method in robot manipulation. Current BC approaches often enhance generalization by leveraging large datasets and incorporating additional visual and textual modalities to capture more diverse information. However, these methods overlook whether the learned representations contain redundant information and lack a solid theoretical foundation to guide the learning process. To address these limitations, we adopt an information-theoretic perspective and introduce mutual information to quantify and mitigate redundancy in latent representations. Building on this, we incorporate the Information Bottleneck (IB) principle into BC, which extends the idea of reducing redundancy by providing a structured framework for compressing irrelevant information while preserving task-relevant features. This work presents the first comprehensive study on redundancy in latent representations across various methods, backbones, and experimental settings, while extending the generalizability of the IB to BC. Extensive experiments and analyses on the CortexBench and LIBERO benchmarks show consistent performance improvements with IB across various settings, underscoring the importance of reducing input data redundancy and highlighting its practical value for real-world applications. Project Page: BC-IB Website.

1. Introduction

Behavior Cloning (BC), one of the simplest and most widely used methods in Imitation Learning (IL), learns a mapping from states to actions by training on state-action pairs from expert demonstrations. BC has been widely studied in autonomous driving (Bain & Sammut, 1995; Torabi et al., 2018), robotics control (Argall et al., 2009) and game AI (Pearce & Zhu, 2022). In robot manipulation, BC has become a foundational approach, enabling robots to replicate expert actions based on sensory inputs such as images or proprioception information like gripper states. To enhance the generalization of robots, most BC methods focus on incorporating large datasets of human or manipulation videos (Jang et al., 2022; Karamcheti et al., 2023; Brohan et al., 2023; Cheang et al., 2024; Saxena et al., 2025), or integrating additional text and visual information (Jia et al., 2024; Wen et al., 2024; Hu et al., 2024). While these methods have made significant progress in improving generalization by leveraging more diverse information, they often neglect a critical aspect: whether the learned representations contain significant redundant information.

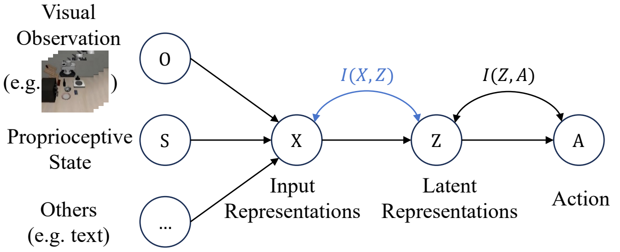


Figure 1: Policy architecture of BC. Current BC methods (black arrows) do not impose restrictions on the latent representations Z, potentially allowing redundant information from the input representations X.

Why do we need to explore this? Firstly, the inherent challenges of input data redundancy remain largely unexplored in BC for robot manipulation, despite their potential impact on policy performance and generalization. Secondly, most existing methods lack a solid theoretical foundation to guide the learning process. This raises a key question: how can we formally characterize and reduce redundancy in inputs or representations in a theoretically grounded way?

How to explore this? As illustrated in Figure 1, in BC, the inputs are typically encoded into individual representations and concatenated to form the input representation X. This is

[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China. [2] Westlake University [3] Zhejiang University. Correspondence to: Badong Chen <chenbd@mail.xjtu.edu.cn>, Donglin Wang <wangdonglin@westlake.edu.cn>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

then processed through a feature fusion module to produce the latent representation Z, which is subsequently decoded to predict the action A. The policy is optimized by minimizing the discrepancy between the predicted actions and the expert-provided actions. In information theory, mutual information between X and Z, denoted as $I(X, Z)$, measures the amount of information gained about one random variable by knowing the other. In BC, if output Y can be well predicted by Z, reducing $I(X, Z)$ means continuously eliminating redundant information from X.

Taking a step further, an information-theoretic approach that balances the trade-off between representation complexity and predictive power offers a *natural framework* to address the problem of latent representation redundancy and the lack of a solid theoretical foundation, namely information bottleneck (IB) principle (Tishby et al., 1999). IB regularizes the representation Z by minimizing the mutual information $I(X, Z)$ between X and Z, while maximizing the mutual information $I(Z, A)$ between Y and A. The first term $I(X, Z)$ represents the compression of the representation, where a smaller mutual information indicates a greater degree of compression and redundancy reduction, while $I(Z, A)$ ensures predictive power is maintained.

Motivated by this information-theoretic approach, we make the first attempt in this work to study the impact of latent representation redundancy in BC for robot manipulation and extend the IB method to this context, where redundancy in latent representations is quantified by $I(X, Z)$. We conduct extensive experiments in various settings and analyses to validate its effectiveness, highlighting the benefits of reducing redundancy to enhance generalization in robotic tasks. Additionally, we provide detailed theoretical analyses, including generalization error bounds, to validate its effectiveness.

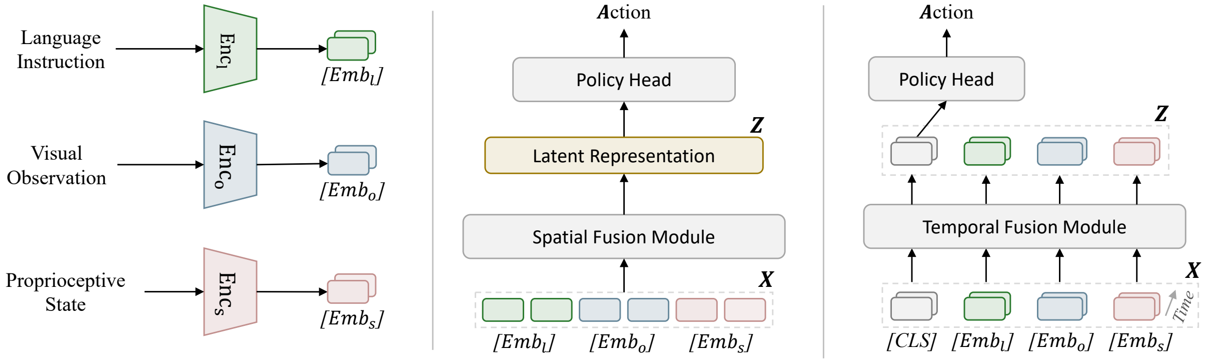
How to apply IB to the BC architectures, and what are its potential applications? To ensure the generality of our findings, we categorize BC architectures based on their feature fusion methods into two types: spatial fusion and temporal fusion. This allows us to identify the applicable scenarios for each fusion method, and by incorporating IB, we uncover a series of interesting findings. Furthermore, our experiments reveal that regardless of the pre-training stage, the final fine-tuning phase, or the size of the dataset, incorporating IB by reducing redundancy enables the model to learn more robust features and improve performance, suggesting its potential applicability in these scenarios.

Our contributions are three-fold. (1) We extend the IB to BC and provide a comprehensive study on the impact of latent representation redundancy in BC for robot manipulation. (2) We empirically demonstrate that minimizing redundancy in latent representations helps existing BC algorithms significantly improve generalization performance on the Cortexbench and LIBERO benchmarks across various settings, indirectly highlighting the considerable redundancy present in current robot trajectory datasets. (3) We provide a detailed theoretical analysis explaining why IB enhances the transferability of BC methods.

2. Related Work

Behavior Cloning in Robot Manipulation. Behavior Cloning (BC), first introduced by (Pomerleau, 1991), is a well-known Imitation Learning (IL) algorithm that learns a policy by directly minimizing the discrepancy between the agent's actions and those of the expert in the demonstration data. To learn more generalizable representations, one class of visual representation learning methods pre-trains on large video datasets of robotics or humans, enabling rapid application of the pre-trained encoder to downstream robotic tasks. Notable examples include VC-1 (Majumdar et al., 2023), R3M (Nair et al., 2023), and Voltron (Karamcheti et al., 2023) . Meanwhile, another line of research focuses on training on even more extensive and diverse datasets with larger models, such as Internet-scale visual question answering and robot trajectory data (Brohan et al., 2023), as well as a vast collection of Internet videos (Cheang et al., 2024). Additionally, some methods further enhance generalization by incorporating additional sources of information. These include inferring textual descriptions based on the robot's current state (Zawalski et al., 2024), leveraging visual trajectories (Wen et al., 2024) and generated images (Tian et al., 2025), and integrating 3D visual information (Goyal et al., 2023). However, these methods have not deeply analyzed the redundancy in learned latent representations, and most also lack a solid theoretical foundation. Thus we extend the Information Bottleneck (IB) principle to BC, addressing this fundamental gap.

Information Bottleneck in Robotics. The Information Bottleneck (IB) principle was first proposed in (Tishby et al., 1999) within the context of information theory. Since then, it has been widely applied in deep learning and various downstream tasks to balance the trade-off between representation accuracy and complexity, including classification (Federici et al., 2019), segmentation (Bardera et al., 2009; Lee et al., 2021), and generative tasks (Jeon et al., 2021). In robotics learning, IB has found notable applications in reinforcement learning, where some works maximize the mutual information between the representation and the dynamics or value function, while restricting the information to encourage the encoder to extract only task-relevant features (Kim et al., 2019; Bai et al., 2021; He et al., 2024). In imitation learning, it has been applied to alleviate the copycat problem from observation histories (Wen et al., 2020). Different from prior works, we introduce IB into Behavior Cloning to explore and empirically validate the redundancy in latent representations in robotics. Additionally, we demonstrate its effectiveness through detailed theoretical analyses.



(a) Feature Extraction (b) Spatial Fusion (c) Temporal Fusion

Figure 2: Model architectures used in this study. Based on feature fusion methods, we categorize the BC methods in robot manipulation into two types: spatial fusion and temporal fusion. After extracting features from each modality a), spatial fusion b) extracts spatial features at a given time step or concatenates features across multiple time steps using encoders like MLPs or CNNs. Temporal Fusion c) fuses input features by modeling dynamic relationships and dependencies between time steps using RNNs or Temporal Transformers. The latent representations are then decoded into actions via the policy head.

3. Preliminary

3.1. Problem Setting of Behavior Cloning

BC can be formulated as the Markov Decision Process (MDP) framework (Torabi et al., 2018), which is often defined without an explicitly specified reward function, to model sequential action generation problems. The concept of rewards is replaced with supervised learning, and the agent learns by mimicking expert actions. Formally, in robot manipulation, the state at each timestep consists of visual observations o_t, the robot's proprioceptive state s_t, and optionally a language instruction l. Let $x_t = (o_t, s_t, l)$ represent the overall state. The policy π maps a sequence of states to an action: $\hat{a}_t = \pi(x_{t-\tau:t})$, where τ indicates the length of the state history. For simplicity, we set $\tau = 1$. The optimization process can be formulated as:

$$\pi^* = \operatorname{argmin}_\pi \mathbb{E}_{(x_t,a_t)\sim\mathcal{D}_e} \left[\mathcal{L}\left(\pi\left(x_t\right), a_t\right)\right], \tag{1}$$

where $\mathcal{D}_e$ is expert trajectory dataset and a_t is action labels. In vanilla BC, $\mathcal{L}$ typically represents the mean squared error (MSE) loss function for continuous action spaces, or cross-entropy (CE) loss for discrete action spaces. In this study, we adopt the continuous action spaces with MSE loss:

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{(x_t,a_t)\sim\mathcal{D}_e} \left[\left\|\pi\left(x_t\right) - a_t\right\|^2\right]. \tag{2}$$

Building on this vanilla BC loss, some methods also introduce alignment loss (Jang et al., 2022; Ma et al., 2024) and reconstruction loss (Radosavovic et al., 2023; Karamcheti et al., 2023). However, in this study, to more clearly illustrate the relationship with representation redundancy, we focus solely on the vanilla BC loss.

3.2. Mutual Information Neural Estimation

Estimating mutual information between variables directly is challenging, thus we use Mutual Information Neural Estimation (MINE) (Belghazi et al., 2018) to estimate it. MINE is based on neural networks, which can efficiently handle high-dimensional, continuous, discrete, and hybrid data types without requiring assumptions about the underlying distributions. MINE estimates mutual information by training a classifier to differentiate between samples from the joint distribution P_{XZ} and the product of the marginal distributions $P_X \otimes P_Z$ of the random variables X and Z. MINE uses a lower bound for mutual information based on the Donsker-Varadhan representation (Donsker & Varadhan, 1983) of the Kullback-Leibler (KL) divergence:

$$\begin{aligned} \mathcal{I}(X;Z) &:= \mathcal{D}_{KL}(P_{XZ}||P_X \otimes P_Z) \geq \widehat{\mathcal{I}}_\theta^{(DV)}(X;Z) \\ &:= \mathbb{E}_{P_{XZ}}\left[T_\theta(x,z)\right] - \log \mathbb{E}_{P_X \otimes P_Z}\left[e^{T_\theta(x,z)}\right], \end{aligned} \tag{3}$$

where $T_\theta : \mathcal{X} \times \mathcal{Z} \to \mathcal{R}$ is a discriminator function modeled by a neural network with parameters θ. We empirically sample from P_{XZ}, and for $P_X \otimes P_Z$, we shuffle the samples from the joint distribution along the batch axis.

4. Pipeline of BC with IB

4.1. Model Architecture

Before introducing IB, we first define its input and latent representations. Traditional IB methods (Amjad & Geiger, 2019; Pacelli & Majumdar, 2020; Wan et al., 2021) typically apply the bottleneck to a single modality (e.g., images or states) and their corresponding latent features, following the information flow $O \to Z \to A$. In contrast, BC for robot manipulation is more complex than earlier control or single-modal tasks, as it requires models to process diverse, multimodal data. This data not only includes RGB images but may also incorporate the robot's proprioceptive state, language instructions, and other modalities, making effective feature fusion essential. Strictly adhering to the conventional IB paradigm would involve constraining each input modality and its corresponding features separately, resulting in a pipeline that is inelegant, difficult to scale, overly complex, and unable to capture cross-modal associations. Furthermore, previous work has shown that proprioceptive states can lead to overfitting (Wang et al., 2024).

As a result, we do not treat image or other modalities separately as inputs to IB, as done in previous studies. Instead, we concatenate features extracted from all modalities through respective feature extractors as our input X, i.e.,

$$x_t = \text{concat}(\text{Enc}_\text{o}(o_t), \text{Enc}_\text{s}(s_t), \text{Enc}_\text{l}(l)), \tag{4}$$

where $\text{Enc}_{(\cdot)}$ denotes the feature extractor of each modality. This results in the information flow $O \to X \to Z \to A$, and brings several advantages: (1) It enables unified redundancy reduction across all modalities. (2) In practice, encoders are often frozen, and this approach is more effective under such conditions. (3) It scales better to a broader range of robotic algorithms. Then, regarding how to process the input X, or how to fuse information from multiple modalities into latent representations Z, we categorize BC methods in robot manipulation into two types based on their feature fusion strategies: spatial fusion and temporal fusion.

As illustrated in Figure 2 (b), spatial fusion involves extracting spatial features from data at a given time step or concatenating features across multiple time steps along the feature dimensions. This approach does not explicitly differentiate between time steps but instead processes the aggregated features as a whole, emphasizing the modeling of inter-feature relationships. The spatial fusion module can be implemented using Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Spatial Transformers, or even simple concatenation operations. These methods are primarily designed to learn highly generalizable visual encoders by leveraging large-scale human video datasets. The pretrained encoders are then fine-tuned for downstream robotic tasks (Majumdar et al., 2023; Zeng et al., 2024).

On the other hand, as illustrated in Figure 2 (c), temporal fusion integrates input features by capturing dynamic relationships and dependencies across time steps. This enables the modeling of both long-term and short-term temporal dynamics in sequential data. Temporal fusion modules can be implemented using Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), or Temporal Transformers. These methods are commonly incorporated into approaches that utilize Transformer-based backbones (Wu et al., 2024; Li et al., 2024b; Liu et al., 2024).

The latent representation Z, which integrates both spatial and temporal information, is then passed through a policy head to generate actions. Existing policy heads primarily focus on using MLPs, Gaussian Mixture Model (GMM), and diffusion-based policy (DP) heads (Chi et al., 2023; Reuss et al., 2024). For simplicity and clearer empirical demonstration, we use an MLP as the policy head.

4.2. Behavior Cloning with Information Bottleneck

The Information Bottleneck (IB) principle is an information-theoretic approach aimed at extracting the most relevant information from an input variable X with respect to an output variable, *i.e.*, action A. The central idea is to find a compressed representation Z of X that retains the relevant information needed to predict A, while discarding irrelevant parts of X that do not contribute to predicting A. The relevant information is quantified as the mutual information $I(X; A)$, and the optimal representation Z is the minimal sufficient statistic of X with respect to A. In practice, this can be achieved by minimizing a Lagrangian that balances the trade-off between retaining predictive information and compressing the input, which can be formulated as:

$$\mathcal{L} = \beta I(X; Z) - I(Z; A), \tag{5}$$

where β is the Lagrange multiplier that balances the trade-off between the compression ability and the predictive power. Thus Equation (2) can be modified as:

$$\mathcal{L}_{\text{BC-IB}} = \mathbb{E}_{(x_t,a_t)\sim\mathcal{D}_e}\left[\beta I(x_t, z_t) + \|\pi(x_t) - a_t\|^2\right], \tag{6}$$

where $z_t = F(x_t)$ and $F(\cdot)$ denotes the fusion module. When $\beta < 0$, the information from X to Z increases; when $\beta = 0$, the method reduces to vanilla BC; and when $\beta > 0$, the information from X to Z decreases.

4.3. Theoretical Analysis

We provide a theoretical analysis of our BC-IB objective in Equation (5). We adapt Theorem 4.1 and Theorem 4.2 to reveal that the generalization error is upper-bounded by the mutual information between the input O and the latent representation Z, following the information flow $O \to Z \to A$. Minimizing this mutual information tightens the bound and improves generalization. However, when O is diverse and multimodal, directly minimizing the mutual information between each modality and its corresponding Z is computationally intractable and unnecessarily complex. To address this, we extract and concatenate features from all modalities into an intermediate feature X, obtain Z via a fusion network f, and minimize the mutual information between X and Z instead. To validate the compatibility of this paradigm with the original theorems, we present Theorem 4.3. The theorem establishes that, even if we optimize $I(X; Z)$ by applying the bottleneck at an intermediate feature level X, as long as X preserves the essential structure of the original input O, we are effectively controlling $I(O;Z)$, with the difference bounded by a small constant δ.

Theorem 4.1. *Generalization Bound Adapted from (Shwartz-Ziv et al., 2019). Let $S = \{(x_t, a_t)\}_{t=1}^{n}$ denote the training data sampled from the same distribution as the random variable pair (X, A). Given the policy π trained on S, the generalization error is given by:*

$$\Delta(S) = \mathbb{E}_{X,A}[\ell(\pi(X), A)] - \frac{1}{n}\sum_{t=1}^{n} \ell(\pi(x_t), a_t). \tag{7}$$

Using the Probably Approximately Correct (PAC) bound framework and the Asymptotic Equipartition Property (AEP) (Cover, 1999), with probability at least $1-\delta$, the following upper bound on the generalization error holds:

$$\Delta(S) \le \sqrt{\frac{2I(X;Z) + \log\frac{2}{\delta}}{2n}}, \tag{8}$$

where $I(X;Z)$ represents the mutual information between the input X and the intermediate representation Z, and δ is the confidence level. Details of proof can be seen in Appendix A of (Shwartz-Ziv et al., 2019).

Theorem 4.2. *Generalization Bound Adapted from (Kawaguchi et al., 2023). Let $S = \{(x_t, a_t)\}_{t=1}^{n}$ denote the training data sampled from the same distribution as the random variable pair (X, A). The generalization error is approximately bounded by:*

$$\Delta(S) \propto \sqrt{\frac{I(X; Z \mid A) + I(\phi^S; S)}{n}}, \tag{9}$$

where ϕ^S is the encoder mapping the input X to the intermediate representation Z. This bound indicates that the generalization error is:

- *Positively correlated with $I(X; Z \mid A)$, which captures mutual information between the input X and the latent representation Z, conditioned on the actions A. This term reflects that the IB compresses X into Z while preserving the relevant information for predicting A.*
- *Positively correlated with $I(\phi^S; S)$, which reflects the information content of the representation ϕ for the given dataset S.*

Theorem 4.3. *Optimization Gap under Different Input Compression. Let $o \to x \to z$ form a Markov chain, where o is transformed into x by a network f, and x is further transformed into z by a network ϕ. Let $\phi_o = f \circ \phi$. Define two optimization problems:*

$$(\theta^\varepsilon, \phi_o^\varepsilon) = \arg\min_{\theta,\phi_o} \mathbb{E}_{P_{\phi_o}(o,x,z)}\left[\log\frac{P_\phi(z|x)}{P_\phi(z)} - \frac{1}{\beta}J(z;\theta)\right], \tag{10}$$

$$(\theta^\star, \phi_o^\star) = \arg\min_{\theta,\phi_o} \mathbb{E}_{P_{\phi_o}(o,z)}\left[\log\frac{P_{\phi_o}(z|o)}{P_{\phi_o}(z)} - \frac{1}{\beta}J(z;\theta)\right]. \tag{11}$$

Let $J^\varepsilon = \mathbb{E}_{P_{f^\varepsilon,\phi^\varepsilon}(o,x,z)}[J(z;\theta^\varepsilon)]$, $J^\star = \mathbb{E}_{P_{\phi_o^\star}(o,z)}[J(z;\theta^\star)]$. Assume the mutual information gap satisfies the following condition: for any δ, we have

$$I(o,z;\phi_o^\varepsilon) - I(o,z;\phi_o^*) \le \frac{\delta}{\beta}. \tag{12}$$

Then, the gap between the two optimizations is bounded as:

$$|J^\star - J^\varepsilon| \le \delta. \tag{13}$$

The detailed proof can be found in Appendix A.

5. Experiments

5.1. Embodied Evaluation

Simulation Benchmarks. We mainly evaluate BC with IB across two benchmarks, CortexBench (Majumdar et al., 2023) and LIBERO (Liu et al., 2024). CortexBench is a single-task benchmark. For validation, we selected four imitation learning-related simulators, encompassing a total of 14 tasks: Adroit (2 tasks) (Rajeswaran et al., 2018), Meta-World (5 tasks) (Yu et al., 2020), DMControl (5 tasks) (Tassa et al., 2018), and TriFinger (2 tasks) (Wuthrich et al., 2021). During evaluation, the number of validation trajectories is set to 25, 10, 25, and 25, respectively. LIBERO is a language-conditioned multi-task benchmark. For evaluation, we select four suites: LIBERO-Goal (10 tasks), LIBERO-Object (10 tasks), LIBERO-Spatial (10 tasks), and LIBERO-Long (10 tasks), each focusing on the controlled transfer of knowledge related to task goals, objects, spatial information, and long-horizon tasks, respectively. During evaluation, the number of validation trajectories is set to 20.

Real-world Evaluation. As shown in Figure 4, our real-world experiments use a 6-DOF UR5 arm equipped with a Robotiq 2F-85 gripper and a RealSense L515 base camera for RGB image capture. Following the simulation setup, we evaluate both a single-task setting and a more challenging language-conditioned multi-task setting. The latter introduces increased distractor objects, randomized object positions, and unseen instances during evaluation to assess generalization. We design two tabletop manipulation tasks: Pick, where the robot lifts an object from the table, and Put (Pick and Place), where the robot picks up an object and places it into a bowl. Demonstrations are collected using a 3D mouse with only the base camera. In the single-task setting, we use 25 demonstrations for Pick and 50 for Pick-and-Place. In the multi-task setting, we collect 800 demonstrations in total, with 200 per task. During evaluation, each task is tested over 10 trajectories.

Baselines. In CortexBench, we evaluate four visual imitation learning models: R3M (Nair et al., 2023), Voltron (Karamcheti et al., 2023), VC-1 (Majumdar et al., 2023), and MPI (Zeng et al., 2024). Following the original papers, we use pre-trained models with frozen image encoders for downstream tasks. Additionally, we introduce two full fine-tuning baselines by replacing the encoders with partially uninitialized ResNet-18 (He et al., 2016) and ViT-S (Dosovitskiy, 2021), denoted as ResNet and ViT, respectively. All methods use the two fusion techniques from Section 4.1: an MLP for spatial fusion and a Temporal Transformer for temporal fusion. In LIBERO, we implement four vision-language policy networks. One of them uses a spatial fusion approach, which employs ResNet as the image encoder and an MLP as the fusion module, referred to as BC-MLP. The other three use temporal fusion. Following the original paper, we rename them based on the combination of the image encoder and fusion module: BC-RNN, BC-Transformer, and BC-VILT (Liu et al., 2024). The policy head for all methods is fixed as an MLP. In real-world evaluation, we adopt VC-1 (Majumdar et al., 2023) for the single-task setting and CogAct (Li et al., 2024a) for the language-conditioned multi-task setting. Notably, all baselines with IB are referred to as BC+IB.

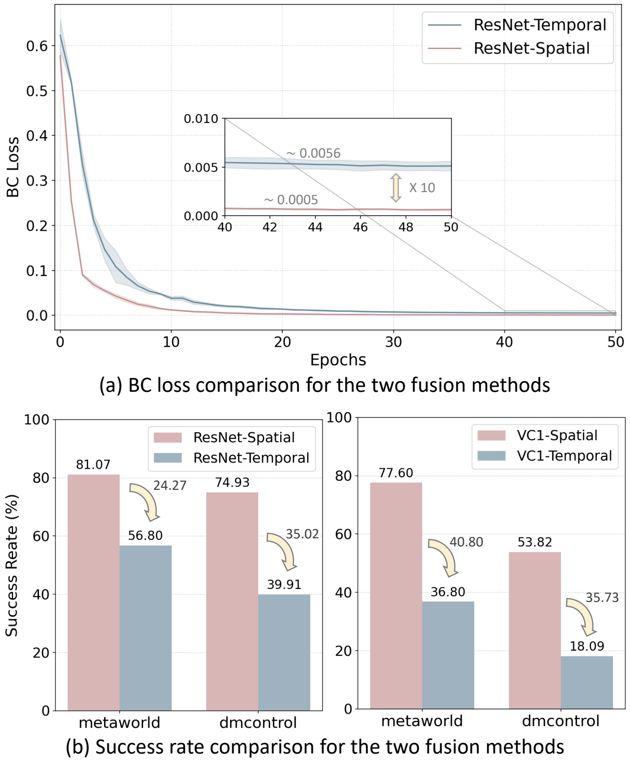
Implementation. In CortexBench, for four partial fine-tuning methods, we train for 100 epochs on each task using the Adam optimizer with a learning rate of 1e-3, a batch size of 512, and weight decay of 1e-4, with learning rate decay applied using a cosine annealing schedule. For two full fine-tuning methods, we train for 50 epochs with a learning rate of 1e-4 and a batch size of 256. In LIBERO, we train for 50 epochs using the AdamW optimizer with a learning rate of 1e-4 and a batch size of 64, decayed using a cosine annealing schedule. In real-world evaluation, we train VC-1 for 200 epochs using the Adam optimizer with a learning rate of 1e-3 and a batch size of 512. CogAct is trained for 8k steps with the AdamW optimizer, using a learning rate of 2e-5 and a batch size of 128. For BC+IB methods, the model used in MINE consists of a two-layer MLP, with a learning rate of 1e-5. The Lagrange multiplier in Equation (6) ranges from 1e-4 to 1e-2 in this work.

Model Selection. For the single-task benchmark CortexBench, we test the model every 5 or 10 epochs and select the model with the highest success rate. For the multi-task benchmark LIBERO, we select the model from the final epoch. For real-world evaluation, we follow the corresponding strategy for each setting as described above.

The appendix provides detailed descriptions of each benchmark (Appendix B.1), all baselines (Appendix B.2), implementation details (Appendix B.3), and the rationale behind the model selection (Appendix B.4).

5.2. Performance on Cortexbench

The Selection of Fusion Method. We first evaluate the effectiveness of the two fusion methods in the baselines on CortexBench, with the results shown in Figure 3.



(a) BC loss comparison for the two fusion methods

(b) Success rate comparison for the two fusion methods

Figure 3: (a) BC loss variation for ResNet in spatial and temporal fusion methods on the bin-picking task of the Meta-World. (b) Averaged success rates of ResNet and VC1 in spatial and temporal fusion methods across the Meta-World and DMControl.

Finding 1: For simple single-task scenarios, spatial fusion is more efficient and effective than temporal fusion. As shown in Figure 3 (b), the performance of methods with temporal fusion drops significantly. From Figure 3 (a), this can be attributed to the slower loss reduction in methods using temporal fusion, which results in higher loss at the same training epoch. Therefore, we focus exclusively on presenting the results for methods employing spatial fusion.

Results. We next report the performance, *i.e.,* success rate, of the baselines and baselines with IB on the single-task benchmark CortexBench in Table 1 with a full-shot setting. Based on results, we derive the following findings.

Finding 2: Whether using full fine-tuning or partial fine-tuning, all vanilla BC methods with different visual backbones incorporating IB outperform their vanilla counterparts across the board. In some benchmarks, the improvements are substantial. For example, ResNet with IB achieves a 10.01% improvement on DMControl, and VC-1 with IB shows a 4.80% improvement on Meta-World. In Appendix C.1, we report the success rate for each task, where

Table 1: Performance in spatial fusion on single-task benchmark CortexBench. We evaluated 14 tasks across 4 benchmarks using 3 random seeds and reported the average success rate along with the standard deviation. * denotes the use of only a small portion of the original model for feature extraction. The best performance is highlighted in bold.

Method	Image Encoder	Adroit	Meta-World	DMControl	TriFinger	Avg
Full Fine-tuning						
ResNet (He et al., 2016) ResNet+IB	ResNet*	66.00±5.29 **72.00**±2.00	81.07±1.22 **83.20**±0.80	74.93±6.21 **84.94**±3.54	71.59±0.88 **72.30**±1.76	73.40 **78.11**
ViT (Dosovitskiy, 2021) ViT+IB	ViT*	35.33±3.06 **37.33**±4.16	31.73±1.67 **36.00**±6.97	10.41±1.21 **12.53**±2.17	55.57±2.65 **55.93**±2.16	33.26 **35.45**
Partial Fine-tuning						
R3M (Nair et al., 2023) R3M+IB	ViT-S	25.33±6.43 **27.33**±3.06	53.07±1.67 **54.13**±2.44	40.31±0.65 **41.74**±5.54	59.87±0.78 **60.63**±0.53	44.65 **45.96**
Voltron (Karamcheti et al., 2023) Voltron+IB	ViT-S	18.67±6.11 **21.33**±5.77	72.53±1.22 **74.40**±3.49	25.35±2.81 **33.16**±6.70	74.21±2.61 **75.12**±2.47	47.69 **51.00**
VC-1 (Majumdar et al., 2023) VC-1+IB	ViT-B	24.67±7.02 **26.00**±9.17	77.60±2.88 **82.40**±2.88	53.82±5.03 **54.93**±1.11	72.05±2.17 **73.80**±1.27	57.04 **59.28**
MPI (Zeng et al., 2024) MPI+IB	ViT-S	34.67±4.16 **36.67**±6.11	66.40±2.12 **69.33**±1.67	59.45±1.91 **61.41**±3.15	61.91±0.57 **63.34**±1.52	55.61 **57.69**

Table 2: Performance on language-condition multi-task benchmark LIBERO. We evaluated 40 tasks of 4 suites using 3 random seeds and reported the average success rate along with the standard deviation. S-Trans. denotes Spatial Transformer and T-Trans. denotes Temporal Transformer. The best performance is bolded.

Method	Image Encoder	Fuse Module	LIBERO-Goal	LIBERO-Object	LIBERO-Spatial	LIBERO-Long	Avg
BC-MLP BC-MLP+IB	ResNet	MLP	16.50±3.97 **27.67**±12.00	19.00±12.22 **31.50**±10.83	29.33±9.61 **41.00**±8.32	2.33±0.76 **2.67**±0.76	16.79 **25.71**
BC-RNN BC-RNN+IB	ResNet	RNN	15.17±10.91 **26.00**±3.50	13.33±7.91 **17.67**±5.77	30.67±13.34 **35.17**±9.45	2.33±0.67 **3.00**±0.17	15.38 **20.46**
BC-Trans. BC-Trans.+IB	ResNet	T-Trans.	67.83±10.42 **74.17**±5.75	41.83±1.89 **45.67**±4.31	68.00±1.00 **72.50**±10.26	15.83±2.52 **18.00**±6.38	48.37 **52.59**
BC-VILT BC+VILT+IB	S-Trans.	T-Trans.	76.17±3.01 **83.83**±3.40	43.00±3.91 **52.00**±3.04	67.17±2.25 **70.67**±2.52	6.50±0.87 **8.67**±1.53	48.21 **53.79**

significant improvements can be observed in certain tasks.

Finding 3: Finding 2 implicitly suggests that the latent representation Z derived from input X is redundant. Therefore, compressing information from input is essential, which can further enhance performance.

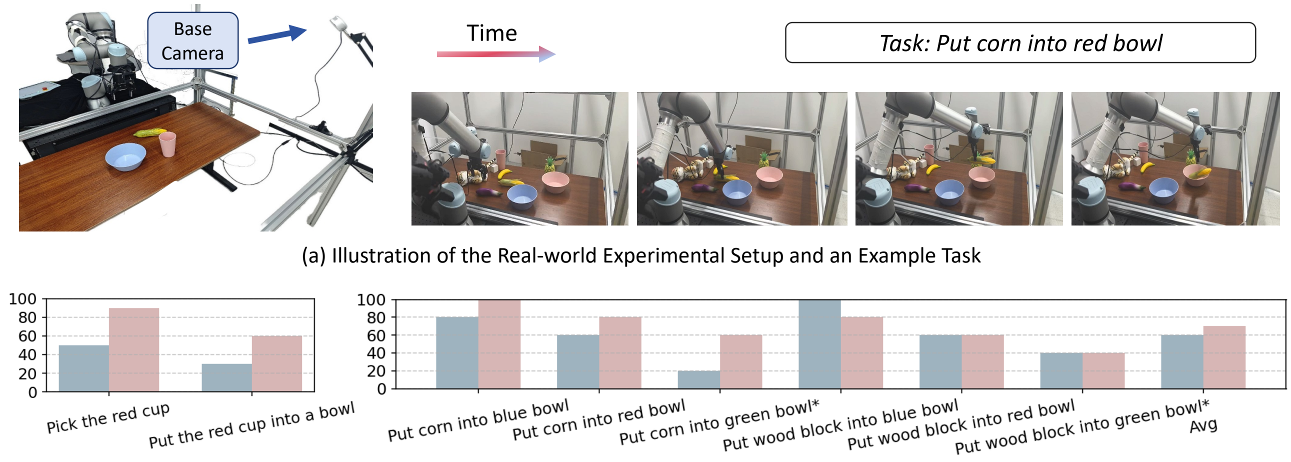
Finding 4: In some benchmarks, particularly Trifinger, the improvement is minimal. We attribute this to the benchmark itself containing very limited redundancy in the visual input, which consists primarily of the robot arm and a single object.

Finding 5: For simple single-task downstream tasks, full fine-tuning of a simple, uninitialized model (ResNet) is sufficient and may even outperform a pre-trained larger model. However, the latter is more efficient for faster fine-tuning and deployment, and proves to be more effective for more complex tasks (Burns et al., 2023).

5.3. Performance on LIBERO

Results. We report the performance on the multi-task benchmark LIBERO with a full-shot setting in Table 2.

Finding 6: For more complex language-conditioned multi-task scenarios, all baselines with different backbones incorporating IB consistently show performance improvements across all LIBERO benchmarks. For example, BC-VILT achieves large gains of 7.66% and 9.00% on LIBERO-Goal and LIBERO-Object, respectively, while BC-RNN shows a significant improvement of 10.83% on LIBERO-Goal. IB proves to be more effective in more complex environments and settings. We attribute this to the difference in task complexity: in CortexBench, the history length is 3, while in LIBERO, it is 10, with LIBERO being a multi-task benchmark and CortexBench being a single-task benchmark. The increased data complexity (task quantity and input informa-



(a) Illustration of the Real-world Experimental Setup and an Example Task

(b) Success Rate in Single-task Setting

(c) Success Rate in Language-conditioned Multi-task Setting

Figure 4: Real-world robot experiments conducted on a tabletop setup with two settings. (a) Left: the experimental setup. (a) Right: an example of predicted trajectories alongside policy execution. (b) and (c): quantitative evaluation results across two settings, where blue denotes the vanilla BC method and red denotes the method with IB. * denotes the unseen tasks.

tion) suggests a higher level of data redundancy, making IB even more effective.

Finding 7: We observe that in complex multi-task scenarios with more intricate inputs, such as a greater number of input modalities and extended historical information, using the Temporal Transformer in temporal fusion proves to be more effective than both spatial fusion and RNN-based temporal fusion. The evidence lies in the fact that the average success rates of BC-Transformer and BC-VILT are over 30% higher than those of BC-MLP and BC-RNN. This is likely because Temporal Transformers excel in handling long-range interactions and capturing dynamic dependencies across time steps, where RNNs and spatial fusion methods may struggle. This finding, together with Finding 1, underscores the specific scenarios in which each fusion method is most applicable.

Finding 8: IB is particularly effective for tasks requiring diverse feature extraction, such as distinguishing distinct task objectives or differentiating between various objects, as in LIBERO-Goal and LIBERO-Object. By filtering out irrelevant information, IB facilitates better generalization and more compact representations. However, the impact is less pronounced in spatial tasks such as LIBERO-Spatial, which rely heavily on structural information that can be disrupted by excessive compression. For long-horizon tasks, the main performance bottleneck lies in the lightweight baseline models, whose limited capacity restricts their effectiveness on more complex tasks such as LIBERO-Long.

5.4. Performance on Real World Experiments

As shown in Figure 4 (b) and (c), incorporating IB consistently improves success rates across both single-task and language-conditioned multi-task real-world settings in most cases. In the single-task setting, VC1+IB significantly outperforms VC1 in both the pick and put tasks. In the more challenging language-conditioned multi-task setting, CogAct+IB consistently outperforms CogAct across most tasks, including unseen object–bowl combinations, demonstrating enhanced generalization capabilities. These results suggest that reducing redundancy in latent representations leads to more robust grasping and more reliable overall execution in real-world scenarios.

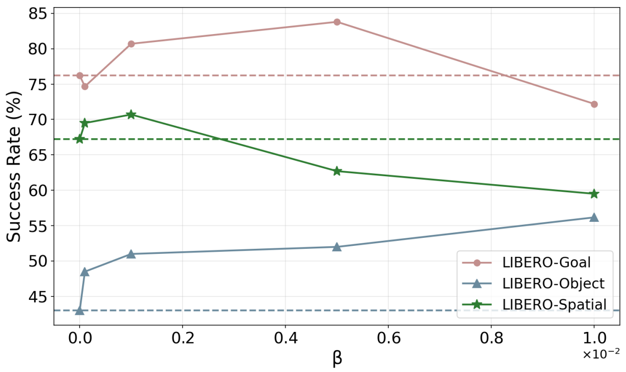


Figure 5: Effect of the Lagrange multiplier β in BC-VILT+IB across three suites of LIBERO. When β=0, the method reduces to vanilla BC-VILT.

5.5. More Analysis

Effect of the Lagrange multiplier β of Equation (6). This experiment evaluates how incorporating IB enhances performance. Since the MINE model's parameters are fixed, the key difference between BC+IB and BC lies in the parameter β, which balances compression and predictive power.

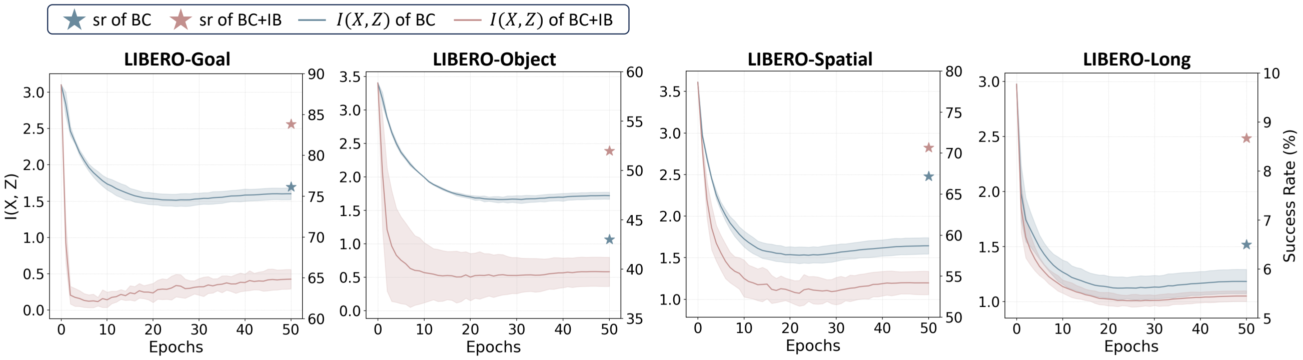


Figure 6: Comparison of vanilla BC and BC+IB on the LIBERO benchmark in terms of success rate (sr) and mutual information. BC-VILT is denoted as BC. BC+IB consistently achieves lower $I(X,Z)$ and higher success rates.

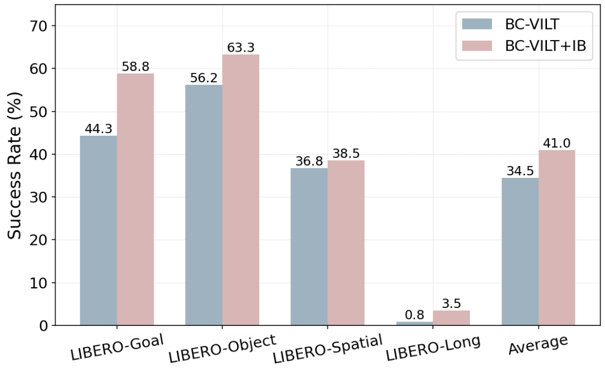


Figure 7: Comparison of the success rates of BC-VILT+IB trained with 10 demonstrations against the vanilla BC-VILT across four LIBERO suites.

For LIBERO experiments, β is explored within 1e-4, 1e-3, 5e-3, 1e-2. As shown in Figure 5, IB improves performance within a specific β range, with a peak observed at an undetermined value. However, across all experiments, β around 1e-4 consistently yields stable improvements.

Effect of the Number of Demonstrations. We evaluate IB's effectiveness in few-shot settings, as few-shot learning is crucial for fine-tuning on domain-specific tasks in real-world applications. As shown in Figure 7, IB consistently improves performance even with limited data across multiple suites in LIBERO, highlighting its effectiveness in real-world scenarios where data is scarce. This further underscores the potential of IB in improving model generalization in practical settings.

Visualizations of $I(X,Z)$. As shown in Figure 6, BC+IB achieves a larger reduction in $I(X,Z)$ compared to vanilla BC, leading to improved performance and validating the effectiveness of IB. For example, in LIBERO-Goal, IB reduces $I(X,Z)$ to one-quarter of its original value and yields a 7.7% increase in success rate.

6. Limitations and Discussion

While our work provides extensive experimental validation of the effectiveness of IB and the necessity of input redundancy reduction in robotics representation learning, several limitations remain. First, we do not comprehensively investigate the scalability of IB in robotic models. Most of our experiments are conducted on relatively lightweight architectures, with only the real-world experiments utilizing CogAct. We have not systematically evaluated large models such as vision-language-action architectures, due to the high computational and time costs involved. Second, alternative policy heads such as transformer-based designs, and models like RT-2 (Brohan et al., 2023) and OpenVLA (Kim et al., 2024), which remove the explicit policy head and instead treat actions as text tokens, have not been explored. We leave these directions for future work. Third, although we evaluate our method on various benchmarks, its robustness to domain shifts, including variations in environment and task settings, remains underexplored. We hope our work will inspire future research and contribute to the ongoing development and refinement of these methods.

7. Conclusions

In this study, we investigated the redundancy in latent representations for behavior cloning in robot manipulation and introduced the Information Bottleneck (IB) principle to mitigate this issue. By incorporating IB, we aimed to filter out redundant information in latent representations while preserving task-relevant features. Extensive experiments across various visual representation learning methods on CortexBench and LIBERO benchmark revealed insightful findings and demonstrated that IB consistently improves performance across diverse tasks and architectures. We hope this work will inspire future research to further integrate information-theoretic principles into robotics, not only as an optimization tool, but also as a framework for theoretical understanding and model design.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. U21A20485), the National Science and Technology Innovation 2030 – Major Project (Grant No. 2022ZD0208800), the NSFC General Program (Grant No. 62176215, 62436005), and the Fundamental Research Funds for Xi'an Jiaotong University (Grant No. xzy022024012).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning in Robotics. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Amjad, R. A. and Geiger, B. C. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2225–2239, 2019.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Bai, C., Wang, L., Han, L., Garg, A., Hao, J., Liu, P., and Wang, Z. Dynamic bottleneck for robust self-supervised exploration. *Advances in Neural Information Processing Systems*, 34:17007–17020, 2021.

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Bardera, A., Rigau, J., Boada, I., Feixas, M., and Sbert, M. Image segmentation using information bottleneck method. *IEEE Transactions on Image Processing*, 18(7): 1601–1612, 2009.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *International conference on machine learning*, pp. 531–540. PMLR, 2018.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2023.

Burns, K., Witzel, Z., Hamid, J. I., Yu, T., Finn, C., and Hausman, K. What makes pre-trained visual representations successful for robust manipulation? *arXiv preprint arXiv:2312.12444*, 2023.

Cheang, C.-L., Chen, G., Jing, Y., Kong, T., Li, H., Li, Y., Liu, Y., Wu, H., Xu, J., Yang, Y., et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Cover, T. M. *Elements of information theory*. John Wiley & Sons, 1999.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Donsker, M. D. and Varadhan, S. S. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on pure and applied mathematics*, 36 (2):183–212, 1983.

Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Federici, M., Dutta, A., Forré, P., Kushman, N., and Akata, Z. Learning robust representations via multi-view information bottleneck. In *International Conference on Learning Representations*, 2019.

Goyal, A., Xu, J., Guo, Y., Blukis, V., Chao, Y.-W., and Fox, D. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18995–19012, 2022.

He, H., Wu, P., Bai, C., Lai, H., Wang, L., Pan, L., Hu, X., and Zhang, W. Bridging the sim-to-real gap from the information bottleneck perspective. In *8th Annual Conference on Robot Learning*, 2024.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In

Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 16000–16009, 2022.

Hu, Y., Guo, Y., Wang, P., Chen, X., Wang, Y.-J., Zhang, J., Sreenath, K., Lu, C., and Chen, J. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024.

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S., and Finn, C. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Jeon, I., Lee, W., Pyeon, M., and Kim, G. Ib-gan: Disentangled representation learning with information bottleneck generative adversarial networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7926–7934, 2021.

Jia, Z., Thumuluri, V., Liu, F., Chen, L., Huang, Z., and Su, H. Chain-of-thought predictive control. In *Forty-first International Conference on Machine Learning*, 2024.

Karamcheti, S., Nair, S., Chen, A. S., Kollar, T., Finn, C., Sadigh, D., and Liang, P. Language-driven representation learning for robotics. In *Robotics: Science and Systems*, 2023.

Kawaguchi, K., Deng, Z., Ji, X., and Huang, J. How does information bottleneck help deep learning? In *International Conference on Machine Learning*, pp. 16049–16096. PMLR, 2023.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Kim, Y., Nam, W., Kim, H., Kim, J.-H., and Kim, G. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International conference on machine learning*, pp. 3379–3388. PMLR, 2019.

Lee, J., Choi, J., Mok, J., and Yoon, S. Reducing information bottleneck for weakly supervised semantic segmentation. *Advances in neural information processing systems*, 34:27408–27421, 2021.

Li, Q., Liang, Y., Wang, Z., Luo, L., Chen, X., Liao, M., Wei, F., Deng, Y., Xu, S., Zhang, Y., et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Li, X., Liu, M., Zhang, H., Yu, C., Xu, J., Wu, H., Cheang, C., Jing, Y., Zhang, W., Liu, H., et al. Vision-language foundation models as effective robot imitators. In *International Conference on Learning Representations*, 2024b.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Ma, T., Zhou, J., Wang, Z., Qiu, R., and Liang, J. Contrastive imitation learning for language-guided multi-task robotic manipulation. In *Conference on Robot Learning*, 2024.

Majumdar, A., Yadav, K., Arnaud, S., Ma, J., Chen, C., Silwal, S., Jain, A., Berges, V.-P., Wu, T., Vakil, J., et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36:655–677, 2023.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, pp. 892–909. PMLR, 2023.

Pacelli, V. and Majumdar, A. Learning task-driven control policies via information bottlenecks. In *Robotics: Science and Systems (RSS)*, 2020.

Pearce, T. and Zhu, J. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, pp. 104–111. IEEE, 2022.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Radosavovic, I., Xiao, T., James, S., Abbeel, P., Malik, J., and Darrell, T. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pp. 416–426. PMLR, 2023.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *Robotics: Science and Systems*, 2018.

Reuss, M., Yağmurlu, Ö. E., Wenzel, F., and Lioutikov, R. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

Saxena, V., Bronars, M., Arachchige, N. R., Wang, K., Shin, W. C., Nasiriany, S., Mandlekar, A., and Xu, D. What matters in learning from large-scale datasets for robot manipulation. In *International Conference on Learning Representations*, 2025.

Shwartz-Ziv, R., Painsky, A., and Tishby, N. REPRESENTATION COMPRESSION AND GENERALIZATION IN DEEP NEURAL NETWORKS, 2019. URL https://openreview.net/forum?id=SkeL6sCqK7.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tian, Y., Yang, S., Zeng, J., Wang, P., Lin, D., Dong, H., and Pang, J. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv preprint physics/0004057*, 1999.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Wan, Z., Zhang, C., Zhu, P., and Hu, Q. Multi-view information-bottleneck representation learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10085–10092, 2021.

Wang, L., Chen, X., Zhao, J., and He, K. Scaling proprioceptive-visual learning with heterogeneous pretrained transformers. In *Advances in neural information processing systems*, 2024.

Wen, C., Lin, J., Darrell, T., Jayaraman, D., and Gao, Y. Fighting copycat agents in behavioral cloning from observation histories. *Advances in Neural Information Processing Systems*, 33:2564–2575, 2020.

Wen, C., Lin, X., So, J., Chen, K., Dou, Q., Gao, Y., and Abbeel, P. Any-point trajectory modeling for policy learning. In *Robotics: Science and Systems*, 2024.

Wu, H., Jing, Y., Cheang, C., Chen, G., Xu, J., Li, X., Liu, M., Li, H., and Kong, T. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Wuthrich, M., Widmaier, F., Grimminger, F., Joshi, S., Agrawal, V., Hammoud, B., Khadiv, M., Bogdanovic, M., Berenz, V., Viereck, J., et al. Trifinger: An open-source robot for learning dexterity. In *Conference on Robot Learning*, pp. 1871–1882. PMLR, 2021.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zawalski, M., Chen, W., Pertsch, K., Mees, O., Finn, C., and Levine, S. Robotic control via embodied chain-of-thought reasoning. In *8th Annual Conference on Robot Learning*, 2024.

Ze, Y., Zhang, G., Zhang, K., Hu, C., Wang, M., and Xu, H. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

Zeng, J., Bu, Q., Wang, B., Xia, W., Chen, L., Dong, H., Song, H., Wang, D., Hu, D., Luo, P., et al. Learning manipulation by predicting interaction. In *Robotics: Science and Systems*, 2024.

Zhu, H., Yang, H., Wang, Y., Yang, J., Wang, L., and He, T. Spa: 3d spatial-awareness enables effective embodied representation. *arXiv preprint arXiv:2410.08208*, 2024.

A. Proof of Theorem 4.3

Proof. The first optimization problem optimizes $I(x, z)$, which imposes a looser constraint on $I(o, z)$, as it does not directly regulate the information flow from o to z. In contrast, the second optimization problem directly constrains $I(o, z)$, which may result in a smaller $I(o, z; \phi_o^\star)$. Therefore, we have:

$$I(o, z; \phi_o^\varepsilon) \geq I(o, z; \phi_o^\star). \tag{14}$$

From the optimization objectives of the two problems, it follows that:

$$I(o, z; \phi_o^\varepsilon) - \frac{1}{\beta} J^\varepsilon \geq I(o, z; \phi_o^\star) - \frac{1}{\beta} J^\star. \tag{15}$$

Rearranging this inequality gives:

$$|J^\varepsilon - J^\star| \leq \beta \cdot \left(I(o, z; \phi_o^\varepsilon) - I(o, z; \phi_o^\star) \right). \tag{16}$$

According to the assumption that the mutual information gap is bounded:

$$I(o, z; \phi_o^\varepsilon) - I(o, z; \phi_o^\star) \leq \frac{\delta}{\beta}, \tag{17}$$

We substitute this bound into the inequality:

$$|J^\varepsilon - J^\star| \leq \beta \cdot \frac{\delta}{\beta} = \delta. \tag{18}$$

Thus, the performance gap is bounded as:

$$|J^\star - J^\varepsilon| \leq \delta. \tag{19}$$

This completes the proof. □

B. Details of Experiment Setting

B.1. Details of Benchmarks

B.1.1. CortexBench

We provide a detailed overview of the four imitation learning benchmarks used in CortexBench (Majumdar et al., 2023). CortexBench is a single-task benchmark that includes 7 selected simulators, collectively offering 17 different embodied AI tasks spanning locomotion, navigation, and both dexterous and mobile manipulation. Three of the simulators are primarily designed for reinforcement learning and are therefore excluded from our analysis. The remaining four simulators, with a total of 14 tasks, are retained for validation: Adroit (2 tasks) (Rajeswaran et al., 2018), Meta-World (5 tasks) (Yu et al., 2020), DMControl (5 tasks) (Tassa et al., 2018), and TriFinger (2 tasks) (Wuthrich et al., 2021).

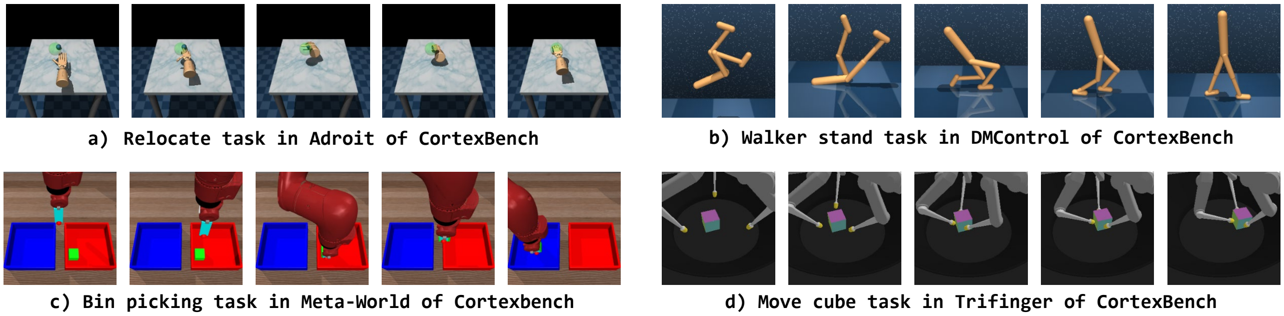
First, Adroit (Rajeswaran et al., 2018) is a suite of dexterous manipulation tasks in which an agent controls a 28-DoF anthropomorphic hand. It includes two of the most challenging tasks: Relocate and Reorient-Pen. In these tasks, the agent must manipulate an object to achieve a specified goal position and orientation. Each task consists of 100 demonstrations.

Second, MetaWorld (Yu et al., 2020) is a collection of tasks in which agents command a Sawyer robot arm to manipulate objects in a tabletop environment. CortexBench includes five tasks from MetaWorld: Assembly, Bin-Picking, Button-Press, Drawer-Open, and Hammer. Each task consists of 25 demonstrations.

Third, DeepMind Control (DMControl) (Tassa et al., 2018) is a widely studied image-based continuous control benchmark, where agents perform locomotion and object manipulation tasks. CortexBench includes five DMC tasks: Finger-Spin, Reacher-Hard, Cheetah-Run, Walker-Stand, and Walker-Walk. Each task consists of 100 demonstrations.

Lastly, TriFinger (TF) (Wuthrich et al., 2021) is a robot consisting of a three-finger hand with 3-DoF per finger. CortexBench includes two tasks from TriFinger: Push-Cube and Reach-Cube. Each task consists of 100 demonstrations.

Although only Meta-World is strictly a robot manipulation benchmark, we include all tasks to demonstrate the effectiveness of IB comprehensively. We provide visualizations for one task from each benchmark, as shown in Figure 8.



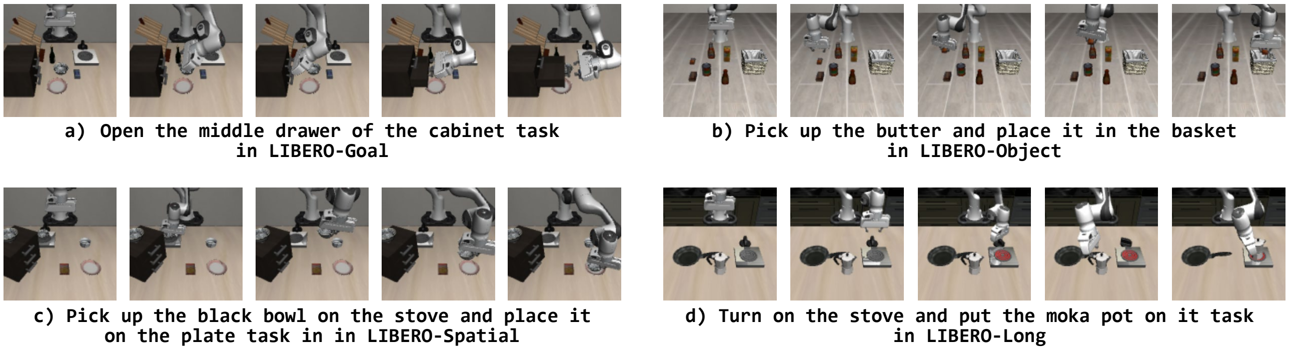
a) Relocate task in Adroit of CortexBench

b) Walker stand task in DMControl of CortexBench

c) Bin picking task in Meta-World of Cortexbench

d) Move cube task in Trifinger of CortexBench

Figure 8: Visualizations for one task from each suite in CortexBench.



a) Open the middle drawer of the cabinet task in LIBERO-Goal

b) Pick up the butter and place it in the basket in LIBERO-Object

c) Pick up the black bowl on the stove and place it on the plate task in in LIBERO-Spatial

d) Turn on the stove and put the moka pot on it task in LIBERO-Long

Figure 9: Visualizations for one task from each suite in LIBERO.

B.1.2. LIEBRO

LIBERO is a language-conditioned multi-task benchmark comprising 130 tasks across five suites. LIBERO (Liu et al., 2024) has four task suites: LIBERO-Goal (10 tasks), LIBERO-Object (10 tasks), LIBERO-Spatial (10 tasks), and LIBERO-100 (100 tasks).

LIBERO-Goal tasks share the same objects with fixed spatial relationships but differ in task goals, requiring the robot to continually acquire new knowledge about motions and behaviors. Examples include (1) opening the middle drawer of the cabinet, (2) opening the top drawer and placing the bowl inside, (3) pushing the plate to the front of the stove, (4) placing the bowl on the plate, (5) placing the bowl on the stove, (6) placing the bowl on top of the cabinet, (7) placing the cream cheese in the bowl, (8) placing the wine bottle on the rack, (9) placing the wine bottle on top of the cabinet, and (10) turning on the stove.

LIBERO-Object tasks involve the robot picking and placing unique objects, requiring it to continually learn and memorize new object types. Examples include (1) picking up the alphabet soup and placing it in the basket, (2) picking up the BBQ sauce and placing it in the basket, (3) picking up the butter and placing it in the basket, (4) picking up the chocolate pudding and placing it in the basket, (5) picking up the cream cheese and placing it in the basket, (6) picking up the ketchup and placing it in the basket, (7) picking up the milk and placing it in the basket, (8) picking up the orange juice and placing it in the basket, (9) picking up the salad dressing and placing it in the basket, and (10) picking up the tomato sauce and placing it in the basket.

LIBERO-Spatial requires the robot to place a bowl, selected from the same set of objects, onto a plate. The robot must continually learn and memorize new spatial relationships. Examples include (1) picking up the black bowl between the plate and the ramekin and placing it on the plate, (2) picking up the black bowl from the table center and placing it on the plate, (3) picking up the black bowl in the top drawer of the wooden cabinet and placing it on the plate, (4) picking up the black bowl next to the cookie box and placing it on the plate, (5) picking up the black bowl next to the plate and placing it on the plate, (6) picking up the black bowl next to the ramekin and placing it on the plate, (7) picking up the black bowl on the cookie box and placing it on the plate, (8) picking up the black bowl on the ramekin and placing it on the plate, (9) picking up the black bowl on the stove and placing it on the plate, and (10) picking up the black bowl on the wooden cabinet and placing it on the plate.

LIBERO-100 consists of 100 tasks involving diverse object interactions and versatile motor skills. It can be divided into LIBERO-10 (10 tasks) and LIBERO-90 (90 tasks), where we use LIBERO-10, also referred to as LIBERO-Long, as our benchmark. LIBERO-Long requires the robot to learn long-horizon tasks, demanding it to plan and execute actions over extended periods to accomplish complex objectives. Examples include (1) turning on the stove and placing the moka pot on it, (2) putting the black bowl in the bottom drawer of the cabinet and closing it, (3) putting the yellow and white mug in the microwave and closing it, (4) putting both moka pots on the stove, (5) putting both the alphabet soup and the cream cheese box in the basket, (6) putting both the alphabet soup and the tomato sauce in the basket, (7) putting the cream cheese box and the butter in the basket, (8) putting the white mug on the left plate and the yellow and white mug on the right plate, (9) putting the white mug on the plate and the chocolate pudding to the right of the plate, and (10) picking up the book and placing it in the back compartment of the caddy.

We provide visualizations for one task from each suite, as shown in Figure 9.

B.2. Details of Baselines

B.2.1. Baselines in CortexBench

In CortexBench, the classification of baselines is primarily based on the visual encoder used.

For full fine-tuning baselines, ResNet (He et al., 2016) and ViT (Dosovitskiy, 2021) are baselines built from the original ResNet-18 and ViT-S models, using only a portion of their architecture and with uninitialized parameters.

For partial fine-tuning baselines, R3M (Nair et al., 2023) pre-trains a ResNet model on human videos (Grauman et al., 2022) using time contrastive learning and video-language alignment. For direct comparison, we use the version reproduced with ViT. VC-1 (Majumdar et al., 2023) pre-trains a ViT using Masked Auto-Encoding (MAE) (He et al., 2022) on a mix of human-object interaction videos, navigation, and the ImageNet (Deng et al., 2009) datasets. Voltron (Karamcheti et al., 2023), a framework for language-driven representation learning from human videos and associated captions, pre-trains a ViT using MAE. MPI (Zeng et al., 2024), a framework for interaction-oriented representation learning, directs the model to predict transition frames and detect manipulated objects using keyframes as input. It learns from human videos and associated captions.

If a proprioceptive state is available, it is first transformed into embeddings using a linear layer. Depending on the fusion method, these embeddings are then combined with the visual embeddings. For spatial fusion, an MLP is used, while for temporal fusion, a temporal transformer is employed. The fused features are ultimately processed through an MLP-based policy head to generate actions.

B.2.2. Baselines in LEBERO

Similar to previous work (Zhu et al., 2024), the baselines in LIBERO largely follow the three architectures outlined in the original paper (Liu et al., 2024), which we have renamed as BC-RNN, BC-Transformer, and BC-VILT. These three baselines are part of the temporal fusion methods.

BC-RNN uses a ResNet as the visual backbone to encode per-step visual observations, with an LSTM as the temporal backbone to process a sequence of encoded visual information. The language instruction is incorporated into the ResNet features using the FiLM method (Perez et al., 2018), and is added to the LSTM inputs.

BC-Transformer employs a similar ResNet-based visual backbone but instead uses a transformer decoder (Vaswani, 2017) as the temporal backbone to process outputs from ResNet, which are temporal sequences of visual tokens. The language embedding is treated as a separate token alongside the visual tokens in the input to the transformer.

BC-VILT utilizes a ViT as the visual backbone and a transformer decoder as the temporal backbone. The language embedding is treated as a separate token in the inputs of both the ViT and the transformer decoder. All temporal backbones output a latent vector at each decision-making step.

Additionally, we introduce a spatial fusion method, BC-MLP, which uses a similar ResNet-based visual backbone. The visual and language embeddings are directly concatenated and input into an MLP for fusion. After feature fusion, all methods use an MLP-based policy head to generate actions.

Table 3: Training hyperparameters of all (full | partial) fine-tuning baselines in CortexBench.

Hyperparameters	Training
epoch	50 | 100
batch size	256 | 512
optimizer	AdamW
learning rate	1e-4 | 1e-3
weight decay	1e-4
lr scheduler	Cosine
lr warm up	0
clip grad	100
augmentation	Resize, CenterCrop, Normalize
history length	3

Table 4: Training hyperparameters of all baselines in LIBERO.

Hyperparameters	Training
epoch	50
batch size	64
optimizer	AdamW
learning rate	1e-4
weight decay	1e-4
lr scheduler	Cosine
lr warm up	0
clip grad	100
augmentation	Normalize, ColorJitter
history length	10

Table 5: IB-related Hyperparameters of all baselines in both CortexBench and LIBERO.

IB-related Hyperparameters	Training
MINE model	
architecture	4-layer MLP
hidden size	512
output size	1
optimizer	Adam
learning rate	1e-5
loss weight	0.1
IB loss	
Lagrange multiplier β	[1e-4, 1e-2]

B.3. Details of Implementations

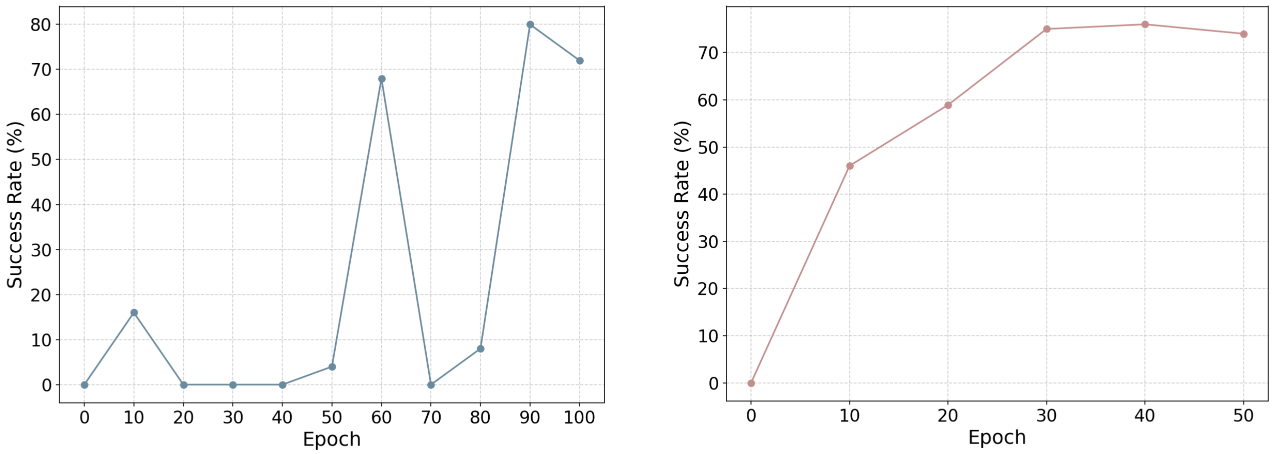
For Cortexbench and LIBERO experiments, we use a single NVIDIA V100 or A100 GPU (CUDA 11.8) with 12 CPUs. For real-world experiments, the single-task setting uses one V100 GPU with 12 CPUs, while the language-conditioned multi-task setting is trained on 8 A100 GPUs with 100 CPUs and evaluated on a single A100 GPU.

B.3.1. CORTEXBENCH

We largely adhere to the original parameter settings from the CortexBench paper (Majumdar et al., 2023). For both full fine-tuning and partial fine-tuning methods, as shown in Table 3, training parameters are presented with full fine-tuning on the left and partial fine-tuning on the right. For model architecture parameters, spatial fusion employs a 4-layer MLP, where the input dimension matches the output dimension of the image encoder. The features are first downsampled and then upsampled to maintain consistency with the input dimension. For temporal fusion, each modality's feature dimension is first projected to 64, then processed through a four-layer, six-head Transformer. For dataset configurations, we adopt a full-shot setting, training with 100 demonstrations for Adroit, 100 for DMControl, 25 for MetaWorld, and 100 for TriFinger. During evaluation, we assess performance using 25, 10, 25, and 25 test trajectories, respectively.

B.3.2. LIBERO

We largely follow the original parameter settings from the LIBERO paper (Liu et al., 2024). The training parameters are provided in Table 4. Regarding model architecture, Appendix A.1 of the original LIBERO paper (Liu et al., 2024) describes the model parameters for BC-RNN, BC-Transformer, and BC-VILT. Here, we present the model parameters for BC-MLP, which shares the same architecture as BC-Transformer except for the fusion module. Specifically, BC-MLP employs a four-layer MLP with a hidden size of 256 as its fusion module. For dataset configurations, in the full-shot setting, we use five demonstrations for evaluation, leaving the remaining 45 for training, which is considered the full-shot setting in



a) Success rate of VC-1 in Assembly task of Meta-World

b) Averaged Success rate of BC-Transformer in LIBERO-Goal

Figure 10: Comparison of success rate curves between single-task and multi-task training.

our experiments. However, full-shot training typically refers to utilizing all 50 demonstrations without allocating any for evaluation, as robotic systems can operate without separate validation data.

For BC+IB, all training and model parameters remain identical to those of BC, except for the IB-specific parameters.

B.4. Details of Model Selection

B.4.1. CORTEXBENCH

In the single-task dataset CortexBench, we observed that the learning curves of certain tasks exhibit significant oscillations, such as the assemble task in MetaWorld, as shown in Figure 10 (a). Previous studies often record performance at intervals of many epochs or steps, selecting either the highest value (Majumdar et al., 2023) or the average of multiple peak values (Ze et al., 2024). Following (Majumdar et al., 2023), we directly use the highest value to explore the model's full potential on the given task.

B.4.2. LIBERO

For the multi-task dataset LIBERO, we observed that while the learning curve for individual tasks may still oscillate, a decrease in success rate for one task is often accompanied by an increase for another. This trade-off results in smoother overall learning curves across multiple tasks, as shown in Figure 10 (b). To make model selection more practical and representative, we directly select the model from the final epoch.

C. Additional Experiment Results

C.1. Details of Simulation Experiments

We provide the task-wise results and corresponding β values for CortexBench. The results for Adroit and TriFinger in Table 6, DMControl in Table 7, and MetaWorld are shown in Table 8. Across almost all tasks in CortexBench, incorporating the IB consistently improves performance compared to vanilla BC methods. Notably, models such as ResNet+IB, VC-1+IB, and MPI+IB often achieve the highest success rates, demonstrating the benefits of redundancy reduction in latent representations. In most cases, properly tuning β (e.g., selecting values in the range of 1e-4 to 1e-2) leads to noticeable improvements.

C.2. Attention Map Visualizations

As shown in Figure 11, IB helps the model focus on task-relevant regions (e.g., the arm and target object) by suppressing attention to redundant background features—an effect less evident without redundancy reduction, highlighting IB's distinct role beyond standard regularization.

Table 6: Task-wise Performance on Adroit and Trifinger of CortexBench. We evaluated 4 tasks of 2 benchmarks using 3 random seeds and reported the average success rate (sr). The best performance is bolded.

		Adroit			TriFinger	
Method	Reorient-Pen	Relocate	Avg	Reach-Cube	Move-Cube	Avg
ResNet	65.33	66.67	66.00	87.12	56.06	71.59
ResNet+IB	**69.33**	**74.67**	**72.00**	**87.14**	**57.45**	**72.30**
ViT	61.33	9.33	35.33	**78.77**	32.37	55.57
ViT+IB	**64.00**	**10.67**	**37.33**	77.83	**34.04**	**55.93**
R3M	45.33	**5.33**	25.33	74.29	45.45	59.87
R3M+IB	**52.00**	2.67	**27.33**	**75.04**	**46.23**	**60.63**
Voltron	32.00	**5.33**	18.67	86.37	62.04	74.21
Voltron+IB	**38.67**	4.00	**21.33**	**86.62**	**63.61**	**75.12**
VC-1	**38.67**	10.67	24.67	84.19	59.90	72.05
VC-1+IB	37.33	**14.67**	**26.00**	**84.69**	**62.91**	**73.80**
MPI	60.00	9.33	34.67	79.69	44.13	61.91
MPI+IB	**61.33**	**12.00**	**36.67**	**79.91**	**46.78**	**63.34**

Table 7: Task-wise Performance on DMControl of CortexBench. We evaluated 5 tasks using 3 random seeds and reported the average success rate (sr). The best performance is highlighted in bold.

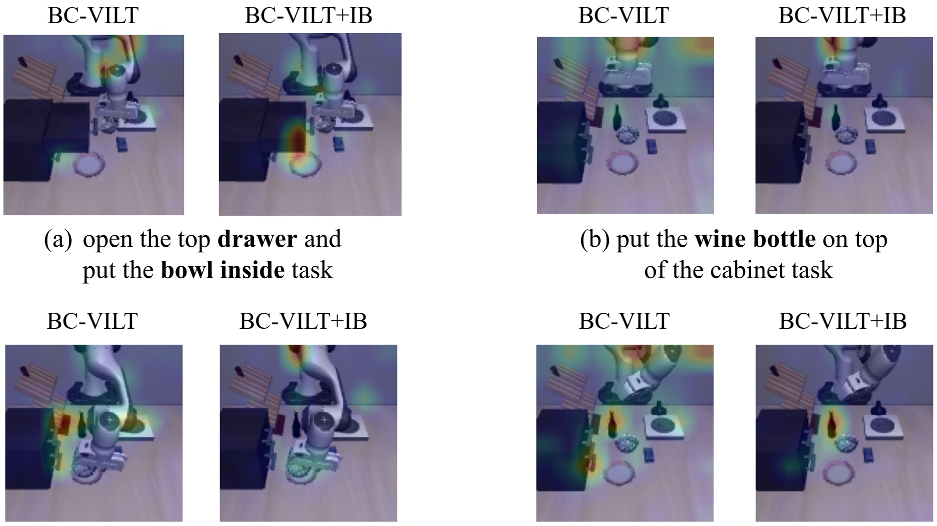
Method	Cheetah-Run	Finger-Spin	Reacher-Easy	Walker-Stand	Walker-Walk	Avg
ResNet	38.32	88.37	92.20	91.42	64.34	74.93
ResNet+IB	**50.75**	**90.42**	**99.78**	**96.39**	**87.37**	**84.94**
ViT	**7.22**	3.39	18.97	**18.05**	4.43	10.41
ViT+IB	4.27	**12.34**	**24.11**	17.45	**4.46**	**12.53**
R3M	**17.01**	65.31	53.73	**49.25**	16.27	40.31
R3M+IB	16.19	**72.10**	**57.34**	46.48	**16.60**	**41.74**
Voltron	1.65	8.56	**44.06**	46.27	26.19	25.35
Voltron+IB	**6.93**	**23.85**	35.89	**56.48**	**42.68**	**33.16**
VC-1	20.02	**85.35**	74.01	64.65	25.07	53.82
VC-1+IB	**21.52**	80.91	**74.80**	**67.10**	**30.34**	**54.93**
MPI	**38.76**	**88.43**	75.87	68.92	25.27	59.45
MPI+IB	33.82	86.44	**86.99**	**69.29**	**30.50**	**61.41**

Table 8: Task-wise Performance on Meta-World of CortexBench. We evaluated 5 tasks using 3 random seeds and reported the average success rate (sr). The best performance is highlighted in bold.

Method	Assembly	Bin-Picking	Button-Press	Drawer-Open	Hammer	Avg
ResNet	40.00	74.67	94.67	100.00	96.00	81.07
ResNet+IB	**49.33**	**76.00**	**94.67**	**100.00**	**96.00**	**83.20**
ViT	13.33	**13.33**	**21.33**	37.33	73.33	31.73
ViT+IB	**13.33**	9.33	18.67	**62.67**	**76.00**	**36.00**
R3M	**42.67**	**56.00**	38.67	66.67	61.33	53.07
R3M+IB	38.67	53.33	**38.67**	**68.00**	**72.00**	**54.13**
Voltron	**60.00**	58.67	**68.00**	82.67	93.33	72.53
Voltron+IB	57.33	**74.67**	54.67	**93.33**	**92.00**	**74.40**
VC-1	68.00	60.00	65.33	100.00	94.67	77.60
VC-1+IB	**70.67**	**76.00**	**69.33**	**100.00**	**96.00**	**82.40**
MPI	61.33	40.00	58.67	100.00	72.00	66.40
MPI+IB	**61.33**	**53.33**	**58.67**	**100.00**	**73.33**	**69.33**

Table 9: Performance on language-condition multi-task benchmark LIBERO. We evaluated 40 tasks of 4 suites using 3 random seeds and reported the average success rate (sr). The best performance is bolded.

Method		LIBERO-Goal	LIBERO-Object	LIBERO-Spatial	LIBERO-10	Avg
BC-MLP	sr	16.50	19.00	29.33	2.33	16.79
BC-MLP+IB	sr	**27.67**	**31.50**	**41.00**	**2.67**	**25.71**
BC-RNN	sr	15.17	13.33	30.67	2.33	15.38
BC-RNN+IB	sr	**26.00**	**17.67**	**35.17**	**3.00**	**20.46**
BC-Transfomer	sr	67.83	41.83	68.00	15.83	48.37
BC-Transfomer+IB	sr	**74.17**	**45.67**	**72.50**	**18.00**	**52.59**
BC-VILT	sr	76.17	43.00	67.17	6.50	48.21
BC+VILT+IB	sr	**83.83**	**52.00**	**70.67**	**8.67**	**53.79**



(a) open the top **drawer** and put the **bowl inside** task

(b) put the **wine bottle** on top of the cabinet task

(c) put the bowl on the **plate** task

(d) put the **wine bottle** on the rack task

Figure 11: IB encourages the model to focus more selectively on task-relevant regions, specifically the robotic arm and the target object, by suppressing attention to redundant background features.

C.3. More Explanation on the LIBERO Experiments

C.3.1. More Experiments on LIBERO-Long

In the main text, we point out that the experimental results on LIBERO-Long are affected by the limited capacity of baseline models (e.g., BC-VILT with 10M parameters), leading to only marginal improvements. While these models perform well on simpler tasks such as LIBERO-Goal, achieving an 80% success rate with an 8% improvement, they show limited gains on more complex tasks like LIBERO-Long, with only a 2% increase in performance. This is primarily due to the performance ceiling imposed by the lightweight baselines. To further illustrate this limitation, we conduct experiments on LIBERO-Long using Diffusion Policy (DP), where the 1.14M-parameter MLP head of BC-Transformer is replaced with the more expressive 90M-parameter DP head. We train DP from scratch on a single A100 GPU with a batch size of 64, a learning rate of 1e-4, and for 50 epochs. The training setup for DP+IB is identical to that of DP, except for the IB-related components. For the IB-specific hyperparameters, the Lagrange multiplier β is set to 1e-5.

Attention: Our experimental setup differs from that of papers like OpenVLA. In those works, the image observations are saved at a resolution of 256×256 (instead of 128×128) and undergo additional filtering, such as removing ”no-op” (zero) actions and unsuccessful demonstrations. In contrast, our setting uses the raw LIBERO data with lower-resolution images and no filtering.

Table 10: Performance on LIBERO-Long.

Method	LIBERO-Long
DP	78.0
DP+IB	**84.0**

C.3.2. Performance on LIBERO-Object

On LIBERO-Object, we observe that the success rate does not consistently improve with an increasing number of demonstrations. Specifically, in the 10-shot setting, BC-VILT achieves a success rate of 56.17%, whereas in the full-shot setting, its performance drops to 43.00%. We hypothesize that this decline is due to inherent data distribution characteristics and suboptimal data quality within the benchmark.

C.4. Extension to Few-shot Setting

We further assess the effectiveness of IB in few-shot settings by evaluating BC-VILT under varying numbers of demonstrations in LIBERO-Goal, as shown in Figure 12. The results consistently show that incorporating IB improves success rates across all LIBERO suites, demonstrating its efficacy in few-shot learning scenarios.

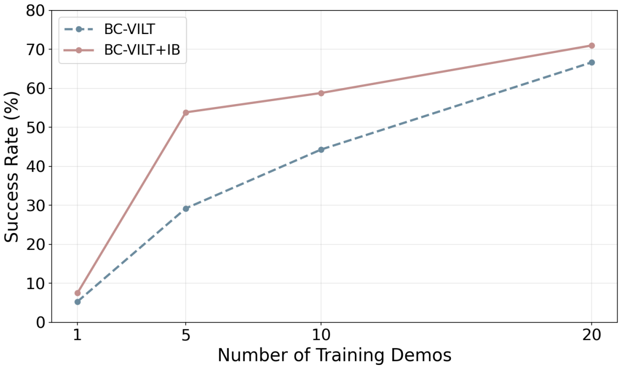


Figure 12: Comparison of the success rates of BC-VILT+IB trained with 1, 5, 10, and 20 demonstrations against the vanilla BC-VILT in the LIBERO-Goal suite.