Drop-Bottleneck: LEARNING DISCRETE COMPRESSED REPRESENTATION FOR NOISE-ROBUST EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel information bottleneck (IB) method named *Drop-Bottleneck*, which discretely drops input features that are irrelevant to the target variable. Drop-Bottleneck not only enjoys a simple and tractable compression objective but also additionally provides a deterministic compressed representation of the input variable, which is useful for inference tasks that require consistent representation. Moreover, it can perform feature selection considering each feature dimension's relevance to the target task, which is unattainable by most neural network-based IB methods. We propose an exploration method based on Drop-Bottleneck for reinforcement learning tasks. In a multitude of noisy and reward sparse maze navigation tasks in VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016), our exploration method achieves state-of-the-art performance. As a new IB framework, we demonstrate that Drop-Bottleneck outperforms Variational Information Bottleneck (VIB) (Alemi et al., 2017) in multiple aspects including adversarial robustness and dimensionality reduction.

1 INTRODUCTION

Data with noise or task-irrelevant information easily harm the training of a model; for instance, the noisy-TV problem (Burda et al., 2019a) is one of well-known such phenomena in reinforcement learning. If observations from the environment are modified to contain a TV screen, which changes its channel randomly based on the agent's actions, the performance of curiosity-based exploration methods dramatically degrades (Burda et al., 2019a;b; Kim et al., 2019; Savinov et al., 2019).

The information bottleneck (IB) theory (Tishby et al., 2000; Tishby & Zaslavsky, 2015) provides a framework for dealing with such task-irrelevant information, and has been actively adopted to exploration in reinforcement learning (Kim et al., 2019; Igl et al., 2019). For an input variable X and a target variable Y, the IB theory introduces another variable Z, which is a compressed representation of X. The IB objective trains Z to contain less information about X but more information about Y as possible, where the two are quantified by mutual information terms of $I(Z;X)$ and $I(Z;Y)$, respectively. IB methods such as variational information bottleneck (VIB) (Alemi et al., 2017; Chalk et al., 2016) and information dropout (Achille & Soatto, 2018) show that the compression of the input variable X can be done by neural networks.

In this work, we propose a novel information bottleneck method named *Drop-Bottleneck* that compresses the input variable by discretely dropping a subset of its input features that are irrelevant to the target variable. Drop-Bottleneck provides some nice properties as follows:

- The compression term of Drop-Bottleneck's objective is simple and is optimized as a tractable solution.
- Drop-Bottleneck provides a *deterministic* compressed representation that still maintains majority of the learned indistinguishability *i.e.* compression. It is useful for inference tasks that require the input representation to be consistent and stable.
- Drop-Bottleneck performs feature selection taking into account each feature dimension's relevance to the target task by learning the feature-wise drop probability. Hence, unlike the compression provided by most neural network-based IB methods, our deterministic representation *reduces* the feature dimensionality, which makes the following inference better efficient with less amount of data.

- Compared to VIB, both of Drop-Bottleneck's original (stochastic) and deterministic compressed representations can greatly improve the robustness to adversarial examples.

Based on the newly proposed Drop-Bottleneck, we design an exploration method that is robust against noisy observations in very sparse reward environments for reinforcement learning. Our exploration maintains an episodic memory and generates intrinsic rewards based on the predictability of new observation from the compressed representations of the ones in the memory. As a result, our method achieves state-of-the-art performance on multiple environments of VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016). We also show that combining our exploration method with VIB instead of Drop-Bottleneck degrades the performance by meaningful margins.

Additionally, we empirically compare with VIB to show Drop-Bottleneck's superior robustness to adversarial examples and ability to reduce feature dimensionality for inference with ImageNet dataset (Russakovsky et al., 2015). We also demonstrate that Drop-Bottleneck's deterministic representation can be a reasonable replacement for its original representation in terms of the learned indistinguishability, with Occluded CIFAR dataset (Achille & Soatto, 2018).

2 Related Work

2.1 Information Bottleneck Methods

There have been a number of IB methods that are approximations to or special forms of the original IB objective. Variational information bottleneck (VIB) (Alemi et al., 2017) approximates the original IB objective by establishing variational bounds on the compression and prediction term. Chalk et al. (2016) propose the same variational bound on the IB objective in the context of sparse coding tasks. The IB theory (Tishby et al., 2000) has been used for various problems that require restriction of information or dealing with task-irrelevant information. Information Dropout (Achille & Soatto, 2018) relates the IB principle to multiple practices in deep learning, including dropout, disentanglement and variational autoencoding. Moyer et al. (2018) obtain representations invariant to specific factors under the variational autoencoder (VAE) (Kingma & Welling, 2013) and VIB frameworks. Amjad & Geiger (2019) discuss the use of IB theory for classification tasks from a theoretical point of view. Dai et al. (2018) employ IB theory for compressing neural networks by pruning neurons in networks. Schulz et al. (2020) propose an attribution method that determines each input feature's importance by enforcing compression of the input variable via the IB framework.

Similar to our goal, some previous research has proposed variants of the original IB objective. Deterministic information bottleneck (DIB) (Strouse & Schwab, 2017) replaces the compression term with an entropy term and solves the new objective using a deterministic encoder. Nonlinear information bottleneck (NIB) (Kolchinsky et al., 2019) modifies the IB objective by squaring the compression term and uses a non-parametric upper bound on the compression term. While DIB is always in the deterministic form, we can flexibly choose the stochastic one for training and the deterministic one for test. Compared to NIB, that is more computationally demanding than VIB due to its non-parametric upper bound, our method is faster.

2.2 Reinforcement Learning with Information Bottleneck Methods

The IB theory has been applied to several reinforcement learning (RL) tasks. Variational discriminator bottleneck (Peng et al., 2019) regulates the discriminator's accuracy using the IB objective to improve adversarial training, and use it for imitation learning. Information Bottleneck Actor Critic (Igl et al., 2019) employs VIB to make the features generalize better and encourage the compression of states as input to the actor-critic algorithm. Curiosity-Bottleneck (Kim et al., 2019) employs the VIB framework to train a compressor that compresses the representation of states, which is still informative about the value function, and uses the compressiveness as exploration signals. InfoBot (Goyal et al., 2019) proposes a conditional version of VIB to improve the transferability of a goal-conditioned policy by minimizing the policy's dependence on the goal. Variational bandwidth bottleneck (Goyal et al., 2020) uses a modified, conditional version of VIB, and solves RL tasks with privileged inputs (*i.e.* valuable information that comes with a cost). Our exploration method differs from these methods in two aspects. First, we propose a new information bottleneck method that is not limited to exploration in RL but generally applicable to the problems for which the IB theory is used. Second, our method generates exploration signals based on *the noise-robust predictability i.e.* the predictability between noise-robust representations of observations.

3 DROP-BOTTLENECK

3.1 PRELIMINARIES OF INFORMATION BOTTLENECK

Given an input random variable X, the information bottleneck (IB) framework (Tishby et al., 2000; Tishby & Zaslavsky, 2015) formalizes a problem of obtaining X's compressed representation Z, which still and only preserves information relevant to the target variable Y. Its objective function is expressed as

$$\text{minimize} -I(Z;Y) + \beta I(Z;X), \tag{1}$$

where β is a Lagrangian multiplier. The first and the second term are prediction and compression terms, respectively. The prediction term $I(Z;Y)$ encourages Z to preserve task-relevant information while the compression term $I(Z;X)$ compresses the input information as much as possible.

As reviewed in previous section, there have been several IB methods (Alemi et al., 2017; Chalk et al., 2016; Achille & Soatto, 2018; Strouse & Schwab, 2017; Kolchinsky et al., 2019), among which the ones derived using variational inference such as variational information bottleneck (VIB) (Alemi et al., 2017) have become dominant due to its applicability to general problems.

3.2 DROP-BOTTLENECK

We propose a novel information bottleneck method called *Drop-Bottleneck* (DB), where the input information is compressed by *discretely* dropping a subset of input features. Thus, its compression objective is simple and easy to optimize. Moreover, its representation is easily convertible to a deterministic version for inference tasks (Section 3.3). While discrete dropping of features has been explored by prior works including Dropout (Srivastava et al., 2014), DB differs in that its goal is to assign different drop probabilities to different feature variables based on their relevance to the target variable.

For an input variable $X = [X_1, \dots, X_d] \in \mathbb{R}^d$ and a drop probability $\boldsymbol{p} = [p_1, \dots, p_d] \in [0,1]^d$, we define its compressed representation as $Z = C_{\boldsymbol{p}}(X) = [c(X_1, p_1), \dots, c(X_d, p_d)] \in \mathbb{R}^d$ such that

$$c(X_i, p_i) = b \cdot \text{Bernoulli}(1-p_i) \cdot X_i, \quad \text{where} \;\; b = \frac{d}{d - \sum_k p_k}, \tag{2}$$

for $i = 1, \dots, d$. That is, the compression procedure *drops* the i-th input feature (*i.e.* replaced by zero) with probability p_i. Since the drop probability is to be learned, we use a scaling factor b to keep the scale of Z constant. We use a single scaling factor for all feature dimensions in order to preserve the relative scales between the features.

With the definition in Equation (2), we derive the compression term of DB to minimize as

$$\begin{aligned} I(Z;X) &= \sum_{i=1}^{d} I(Z_i; X_1, \dots, X_d | Z_1, \dots, Z_{i-1}) && (3) \\ &= \sum_{i=1}^{d} \left[I(Z_i; X_i | Z_1, \dots, Z_{i-1}) + I(Z_i; X_1, \dots, X_d \setminus X_i | Z_1, \dots, Z_{i-1}, X_i) \right] && (4) \\ &= \sum_{i=1}^{d} I(Z_i; X_i | Z_1, \dots, Z_{i-1}) \le \sum_{i=1}^{d} I(Z_i; X_i) = \hat{I}(Z;X) && (5) \end{aligned}$$

using that $Z_i \perp\!\!\!\perp X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_d | Z_1, \dots, Z_{i-1}, X_i$ and $Z_i \perp\!\!\!\perp Z_1, \dots, Z_{i-1} | X_i$. Note that $\hat{I}(Z;X) - I(Z;X) = \left(\sum_{i=1}^{d} H(Z_i) \right) - H(Z_1, \dots, Z_d) = TC(Z)$ where $TC(Z)$ is the total correlation of Z and $H(\cdot)$ denotes the entropy, and $\hat{I}(Z;X) = I(Z;X)$ if $X_1, \dots, X_d$ are independent. To provide a rough view on the gap, due to the joint optimization with the compression term $\hat{I}(Z;X)$ and the prediction term $I(Z;Y)$, Z becomes likely to preserve less redundant and less correlated features, and $TC(Z)$ could decrease as the optimization progresses.

Finally, DB's new compression term, $\hat{I}(Z;X)$, is computed as

$$\hat{I}(Z;X) = \sum_{i=1}^{d} I(Z_i;X_i) = \sum_{i=1}^{d} \big(H(X_i) - H(X_i|Z_i)\big) \tag{6}$$

$$= \sum_{i=1}^{d} \big(H(X_i) - p_i \cdot H(X_i|Z_i = 0) - (1-p_i) \cdot H(X_i|Z_i = bX_i)\big) \tag{7}$$

$$\approx \sum_{i=1}^{d} \big(H(X_i) - p_i \cdot H(X_i) - (1-p_i) \cdot 0\big) = \sum_{i=1}^{d} H(X_i)(1-p_i) \tag{8}$$

From Equation (7) to Equation (8), we use the two ideas: (i) $H(X_i|Z_i = 0) = H(X_i)$ because $Z_i = 0$ means it contains no information about X_i, and (ii) $H(X_i|Z_i = bX_i) = 0$ because $Z_i = bX_i$ means Z_i preserves the feature (*i.e.* Z_i fully identifies X_i) and thus their conditional entropy becomes zero. Importantly, DB's compression term is computed as the simple tractable expression in Equation (8). As the goal of the compression term is to penalize $I(Z;X)$ not $H(X)$, the drop probability $\boldsymbol{p}$ is the only parameter optimized with our compression term. Thus, each $H(X_i)$ can be computed with any entropy estimation method such as the binning-based estimation, since the computation now has no need to be differentiable.

However, one issue of Equation (8) is that Z is not differentiable with respect to $\boldsymbol{p}$ as Bernoulli distributions are not differentiable. We thus use the Concrete relaxation (Maddison et al., 2017; Jang et al., 2016) of the Bernoulli distribution to update $\boldsymbol{p}$ via gradients from Z:

$$\text{Bernoulli}(p) \approx \sigma\left(\frac{1}{\lambda}\left(\log p - \log(1-p) + \log u - \log(1-u)\right)\right), \tag{9}$$

where $u \sim \text{Uniform}(0,1)$ and λ is a temperature for the Concrete distribution. Intuitively, $\boldsymbol{p}$ is trained to assign a high drop probability to the feature that is irrelevant to the target variable Y.

3.3 Deterministic Compressed Representation

With Drop-Bottleneck, one can simply obtain the deterministic version of the compressed representation as $\bar{Z} = \bar{C}_{\boldsymbol{p}}(X) = [\bar{c}(X_1, p_1), \ldots, \bar{c}(X_d, p_d)]$ for

$$\bar{c}(X_i, p_i) = \bar{b} \cdot \mathbb{1}(p_i < 0.5) \cdot X_i, \;\; \text{where} \;\; \bar{b} = \frac{d}{\sum_k \mathbb{1}(p_k < 0.5)}. \tag{10}$$

$\bar{b}$ is defined similarly to b with a minor exception that the scaling is done based on the actual, deterministic number of the dropped features. The deterministic compressed representation $\bar{Z}$ is useful for inference tasks that require stability or consistency of the representation as well as reducing the feature dimensionality for inference, as we demonstrate in Section 5.4.

3.4 Training with Drop-Bottleneck

We present how Drop-Bottleneck (DB) is trained with the full IB objective. Since DB proposes only a new compression term, any existing method for maximizing the prediction term $I(Z;Y)$ can be adopted. We below discuss with an example of Deep Infomax (Hjelm et al., 2019) since our exploration method uses this framework (Section 4). Deep Infomax, instead of $I(Z;Y)$, maximizes its Jensen-Shannon mutual information estimator $I_\psi^{\text{JSD}}(Z;Y)$ defined as

$$I_\psi^{\text{JSD}}(Z;Y) = \frac{1}{2}\left(\mathbb{E}_{\mathbb{P}_{ZY}}[-\zeta(-T_\psi(Z,Y))] - \mathbb{E}_{\mathbb{P}_Z \otimes \mathbb{P}_Y}[\zeta(T_\psi(Z,\tilde{Y}))] + \log 4\right), \tag{11}$$

where T_ψ is a discriminator network with parameter ψ and $\zeta(\cdot)$ is the softplus function.

Finally, the IB objective with Drop-Bottleneck becomes

$$\text{minimize} \; -I_\psi^{\text{JSD}}(Z;Y) + \beta \sum_{i=1}^{d} H(X_i)(1-p_i), \tag{12}$$

which can be optimized via gradient descent. To make $\boldsymbol{p}$ more freely trainable, we suggest simple element-wise parameterization of $\boldsymbol{p}$ as $p_i = \sigma(p_i')$ and initializing $p_i' \sim \text{Uniform}(a,b)$. If the input variable X is obtained by a parametric function, then its parameters can be trained via the prediction term, jointly with $\boldsymbol{p}$ and ψ. In next section, we will discuss its application to hard exploration problems for reinforcement learning.

4 ROBUST EXPLORATION WITH DROP-BOTTLENECK

Based on DB, we propose an exploration method that is robust against highly noisy observations in a very sparse reward environment for reinforcement learning tasks. We first define a parametric feature extractor f_ϕ that maps a state to X. For transitions (S, A, S') where $S, A,$ and S' are current states, actions and next states, respectively, we use the DB objective with

$$X = f_\phi(S'), \quad Z = C_{\boldsymbol{p}}(X), \quad Y = C_{\boldsymbol{p}}(f_\phi(S)). \tag{13}$$

For every transition (S, A, S'), the compression term $I(Z; X) = I(C_{\boldsymbol{p}}(f_\phi(S')); f_\phi(S'))$ encourages $C_{\boldsymbol{p}}$ to drop unnecessary features of the next state embedding $f_\phi(S')$ as possible. The prediction term $I(Z; Y) = I(C_{\boldsymbol{p}}(f_\phi(S')); C_{\boldsymbol{p}}(f_\phi(S)))$ makes the compressed representations of the current and next state embeddings, $C_{\boldsymbol{p}}(f_\phi(S))$ and $C_{\boldsymbol{p}}(f_\phi(S'))$, informative about each other.

Applying Equation (13) to Equation (12), the Drop-Bottleneck objective for exploration becomes

$$\text{minimize} -I_\psi^{\text{JSD}}(C_{\boldsymbol{p}}(f_\phi(S')); C_{\boldsymbol{p}}(f_\phi(S))) + \beta \sum_{i=1}^{d} H((f_\phi(S'))_i)(1 - p_i). \tag{14}$$

While f_ϕ, $\boldsymbol{p}$, and T_ψ are being trained online, we use them for the agent's exploration with the help of episodic memory inspired by Savinov et al. (2019). Starting from an empty episodic memory M, we add the learned feature of the observation at each step. For example, at time step t, the episodic memory is $M = \{\bar{C}_{\boldsymbol{p}}(f_\phi(s_1)), \bar{C}_{\boldsymbol{p}}(f_\phi(s_2)), \ldots, \bar{C}_{\boldsymbol{p}}(f_\phi(s_{t-1}))\}$ where $s_1, \ldots, s_{t-1}$ are the earlier observations from that episode. We then quantify the degree of novelty of a new observation as an intrinsic reward. Specifically, the intrinsic reward for s_t is computed utilizing the Deep Infomax discriminator T_ψ, which is trained to output a large value for joint (or likely) input and a small value for marginal (or arbitrary) input:

$$r^{\text{i}}_{M,t}(s_t) = \frac{1}{t-1} \sum_{j=1}^{t-1} \left[g(s_t, s_j) + g(s_j, s_t) \right], \text{s.t. } g(x, y) = \zeta(-T_\psi(\bar{C}_{\boldsymbol{p}}(f_\phi(x)), \bar{C}_{\boldsymbol{p}}(f_\phi(y))), \tag{15}$$

where $g(s_t, s_j)$ and $g(s_j, s_t)$ denote the unlikeliness of s_t being the next and the previous state of s_j, respectively. Thus, intuitively, for s_t that is close to a region of the state space covered by the earlier observations, $r^{\text{i}}_{M,t}(s_t)$ becomes low, and vice versa. It results in a solid exploration method capable of handling noisy environments with very sparse rewards. For computing the intrinsic reward, we use the DB's deterministic compressed representation of states to provide stable exploration signals to the policy optimization.

5 EXPERIMENTS

We carry out three types of experiments to evaluate Drop-Bottleneck (DB) in multiple aspects. First, we apply DB exploration to multiple VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016) environments with three hard noise settings, where we compare with state-of-the-art methods as well as its VIB variant. Second, we empirically show that DB is superior to VIB for adversarial robustness and feature dimensionality reduction in ImageNet classification (Russakovsky et al., 2015). Finally, in Appendix B, we make another comparison with VIB in terms of removal of task-irrelevant information and the validity of the deterministic compressed representation.

5.1 EXPERIMENTAL SETUP FOR EXPLORATION TASKS

For the exploration tasks on VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016), we use the proximal policy optimization (PPO) algorithm (Schulman et al., 2017) as the base RL method. We employ ICM from Pathak et al. (2017), and EC and ECO from Savinov et al. (2019) as baseline exploration methods. ICM learns a dynamics model of environment and uses the prediction errors as exploration signals for the agent. EC and ECO are curiosity methods that use episodic memory to produce novelty bonus according to the reachability of new observations, and show the state-of-the-art exploration performance on VizDoom and DMLab navigation tasks. In summary, we compare with four baseline methods: PPO, PPO + ICM, PPO + EC, and PPO + ECO. Additionally, we report the performance of our method combined with VIB instead of DB.

We conduct experiments with three specific noise settings named "Image Action", "Noise" and "Noise Action", as proposed in Savinov et al. (2019). We present more details of noise and their observation examples in Appendix D.1. Following Savinov et al. (2019), we resize observations as 160×120 only for the episodic curiosity module while as 84×84 for the PPO policy and exploration methods. We use the official source code[1] of Savinov et al. (2019) to implement and configure the baselines (ICM, EC and ECO) and the three noise settings.

For the feature extractor f_ϕ, we use the same CNN with the policy network of PPO from Mnih et al. (2015). The only modification is to use $d = 128$ *i.e.* 128-dimensional features instead of 512 to make features lightweight enough to be stored in the episodic memory. The Deep Infomax discriminator T_ψ consists of three FC hidden layers with $64, 32, 16$ units each, followed by a final FC layer for prediction. We initialize the drop probability $\boldsymbol{p}$ with $p_i = \sigma(p'_i)$ and $p'_i \sim \text{Uniform}(a, b)$ where $a = -2, b = 1$. We collect samples and update $\boldsymbol{p}, T_\psi, f_\phi$ with Equation (14) every 10.8K and 21.6K time steps in VizDoom and DMLab, respectively, with a batch size of 512. We duplicate the compressed representation 50 times with differently sampled drop masks, which help better training of the feature extractor, the drop probability and the discriminator by forming diverse subsets of features. Please refer to Appendix D for more details of our experimental setup.

5.2 Exploration in Noisy Static Maze Environments

VizDoom (Kempka et al., 2016) provides a static 3D maze environment. We experiment on the *MyWayHome* task with nine different settings by combining three reward conditions ("Dense", "Sparse" and "Very Sparse") in Pathak et al. (2017) and three noise settings in previous section. In the "Dense", "Sparse" and "Very Sparse" cases, the agent is randomly spawned in a near, medium and very distant room, respectively. Thus, "Dense" is a relatively easy task for the agent to reach the goal even with a short random walk, while "Sparse" and "Very Sparse" require the agent to perform a series of directed actions, which make the goal-oriented navigation difficult.

Table 1 and Figure 1 compare the DB exploration with three baselines, PPO, PPO + ICM, and PPO + ECO on the VizDoom tasks, in terms of the final performance and how quickly they learn. The results suggest that even in the static maze environments, the three noise settings can degrade the performance of the exploration by large margins. On the other hand, our method with DB shows robustness against such noise or task-irrelevant information, and outperforms the baselines in all the nine challenging tasks, whereas our method combined with VIB does not exhibit competitive results.

5.3 Exploration in Noisy and Randomly Generated Maze Environments

As a more challenging exploration task, we employ DMLab (Savinov et al., 2019), which are general and randomly generated maze environments where at the beginning of every episode, each maze is procedurally generated with its random goal location. Thanks to the random map generator, each method is evaluated on test mazes that are independent from training mazes. As done in Savinov et al. (2019), we test with six settings according to two reward conditions ("Sparse" and "Very Sparse") and the three noise settings. In the "Sparse" scenario, the agent is (re-)spawned at a random location when each episode begins or every time it reaches the goal *i.e.* the sparse reward; the agent should reach the goal as many times as possible within the fixed episode length. The "Very Sparse" is its harder version: the agent does not get (re-)spawned near or in the same room with the goal.

Table 1 compares between different exploration methods on the DMLab tasks. The results demonstrate that our DB exploration method achieves state-of-the-art performance with significant margins from the baselines on all the 6 tasks, and performs better than our method equipped with VIB as well. The plots too suggest that our method provides stable exploration signals to the agent under different environmental and noise settings. As mentioned in Section 5.1, our method can achieve better performance even using much lower resolution observations of 84×84 than 160×120 of EC and ECO. Also, excluding the policy network, our method maintains 0.5M parameters, which is significantly smaller compared to ECO with 13M parameters.

Figure 2 shows evolution examples of the drop probability distribution over training time steps. It overviews the role of drop probability $\boldsymbol{p}$ in DB. As the joint training of the feature extractor f_ϕ with $\boldsymbol{p}$ progresses, $\boldsymbol{p}$ gets separated into high- and low-value groups, where the former drops task-

[1] https://github.com/google-research/episodic-curiosity.

Table 1: Comparison of the average episodic sum of rewards in VizDoom (over 10 runs) and DMLab (over 30 runs) under three noise settings: Image Action (IA), Noise (N) and Noise Action (NA). The values are measured after 6M and 20M (4 action-repeated) steps for VizDoom and DMLab respectively, with no seed tuning done. Baseline results for DMLab are cited from Savinov et al. (2019). Grid Oracle† provides the performance upper bound by the oracle method for DMLab tasks.

Method	VizDoom									DMLab					
	Dense			Sparse			Very Sparse			Sparse			Very Sparse		
	IA	N	NA	IA	N	NA	IA	N	NA	IA	N	NA	IA	N	NA
PPO (Schulman et al., 2017)	**1.00**	**1.00**	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	8.5	11.6	9.8	6.3	8.7	6.1
PPO + ICM (Pathak et al., 2017)	0.87	**1.00**	**1.00**	0.00	0.50	0.40	0.00	0.73	0.20	6.9	7.7	7.6	4.9	6.0	5.7
PPO + EC (Savinov et al., 2019)	–	–	–	–	–	–	–	–	–	13.1	18.7	14.8	7.4	13.4	11.3
PPO + ECO (Savinov et al., 2019)	0.71	0.81	0.72	0.21	0.70	0.33	0.19	0.79	0.50	18.5	28.2	18.9	16.8	26.0	12.5
PPO + Ours (VIB)	**1.00**	**1.00**	**1.00**	0.21	0.61	0.40	0.37	0.70	0.67	28.2	31.9	27.1	23.5	25.4	22.3
PPO + Ours (Drop-Bottleneck)	**1.00**	**1.00**	**1.00**	**0.90**	**1.00**	**0.99**	**0.90**	**1.00**	**0.90**	**30.4**	**32.7**	**30.6**	**28.8**	**29.1**	**26.9**
PPO + Grid Oracle†	–	–	–	–	–	–	–	–	–	37.4	38.8	39.3	36.3	35.5	35.4

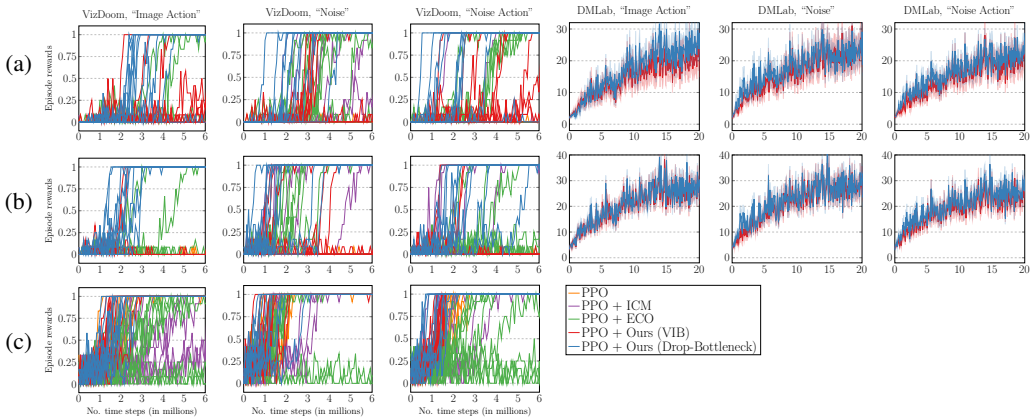


Figure 1: Reward trajectories as a function of training step (in millions) for VizDoom (columns 1-3) and DMLab (columns 4-6) with (a) Very Sparse, (b) Sparse and (c) Dense settings. For VizDoom tasks, we show all the 10 runs per method. For DMLab tasks, we show the averaged episode rewards over 30 runs of our exploration with the 95% confidence intervals.

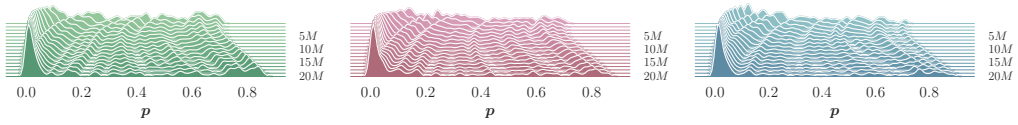


Figure 2: Evolution examples of the drop probability distribution $\boldsymbol{p}$ on Very Sparse DMLab environments with (left) Image Action, (middle) Noise and (right) Noise Action settings. Each figure shows a histogram per $\boldsymbol{p}$ value according to training iterations (the more front is the more recent).

irrelevant features and the latter preserves task-relevant features. This suggests that in the DMLab environments, the DB objective of Equation (14) successfully encourages dropping the features unrelated to transition between observations and also the deterministic compressed representation becomes stable as the training progresses.

5.4 Comparison with VIB: Adversarial Robustness & Dimension Reduction

We experiment image classification on ImageNet (Russakovsky et al., 2015) to compare Drop-Bottleneck (DB) with Variational Information Bottleneck (VIB) (Alemi et al., 2017), which is the most widely-used information bottleneck framework. We compare them in terms of the robustness to adversarial attacks and the reduction of feature dimensionality. We follow the experimental setup suggested in VIB (Alemi et al., 2017), with one minor exception that we use $\beta_1 = 0.9$ and no

Table 2: Results of the adversarial robustness for Drop-Bottleneck (DB) and variational information bottleneck (VIB) (Alemi et al., 2017) with the targeted ℓ_2 and ℓ_∞ attacks (Carlini & Wagner, 2017). *Succ.* denotes the attack success rate in % (lower is better), and *Dist.* is the average perturbation distance over successful adversarial examples (higher is better).

Attack type	β	VIB		DB		DB (determ.)	
		Succ.	Dist.	Succ.	Dist.	Succ.	Dist.
ℓ_2	0.0001	99.5	0.806	100.0	0.929	99.5	0.923
	0.0003162	99.5	0.751	100.0	0.944	100.0	0.941
	0.001	100.0	0.796	99.5	1.097	100.0	1.134
	0.003162	99.5	0.842	27.0	3.434	23.0	2.565
	0.01	100.0	0.936	18.5	6.847	20.0	6.551
	0.03162	100.0	0.695	41.0	2.160	39.5	1.953
	0.1	99.5	0.874	85.5	2.850	85.5	2.348
ℓ_∞	0.0001	99.5	0.015	91.0	0.013	95.5	0.009
	0.0003162	99.5	0.017	85.0	0.016	91.5	0.009
	0.001	100.0	0.017	62.5	0.020	70.0	0.012
	0.003162	97.5	0.017	1.5	0.009	1.5	0.020
	0.01	87.0	0.019	2.0	0.022	2.0	0.013
	0.03162	25.0	0.121	8.5	0.022	8.0	0.023
	0.1	15.5	0.202	23.0	0.017	23.0	0.019

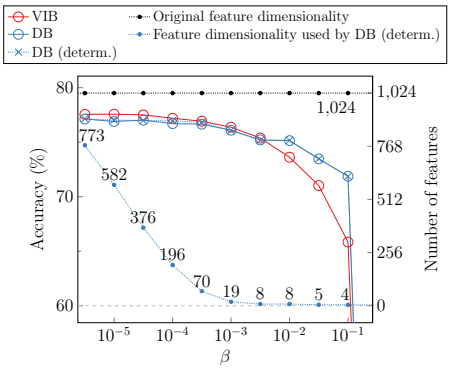


Figure 3: Classification accuracy of Inception-ResNet-v2 equipped with VIB (Alemi et al., 2017) and DB on ImageNet validation set (Russakovsky et al., 2015). DB (determ.) quickly drops many feature dimensions with increased β, while VIB retains them at 1024 regardless of β.

Table 3: Results of the adversarial robustness for Drop-Bottleneck (DB) and the mutual information-based feature selection with the targeted ℓ_2 and ℓ_∞ attacks (Carlini & Wagner, 2017), using the same number of features. *Succ.* denotes the attack success rate in % (lower is better), and *Dist.* is the average perturbation distance over successful adversarial examples (higher is better).

Attack type	# of features	MI-based FS		DB (determ.)	
		Succ.	Dist.	Succ.	Dist.
ℓ_2	196	99.5	1.484	99.5	0.923
	70	100.0	1.323	100.0	0.941
	19	99.5	1.161	100.0	1.134
	8	99.5	1.164	20.0	6.551
	5	97.0	1.202	39.5	1.953
	4	97.0	1.127	85.5	2.348
ℓ_∞	196	99.5	0.016	95.5	0.009
	70	100.0	0.014	91.5	0.009
	19	99.5	0.013	70.0	0.012
	8	99.5	0.014	2.0	0.013
	5	97.0	0.016	8.0	0.023
	4	97.0	0.015	23.0	0.019

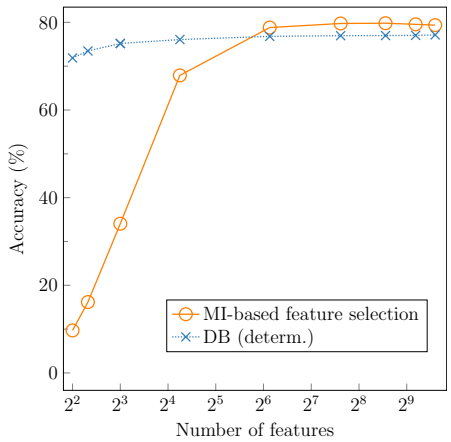


Figure 4: Classification accuracy of Inception-ResNet-v2 equipped with the mutual information-based feature selection and DB on ImageNet validation set (Russakovsky et al., 2015), using the same number of features.

learning rate decay for DB's Adam optimizer (Kingma & Ba, 2015). Additionally, we provide a comparison with the mutual information-based feature selection method in the similar aspects.

Robustness to adversarial attacks. Following Alemi et al. (2017), we employ the targeted ℓ_2 and ℓ_∞ adversarial attacks from Carlini & Wagner (2017). For each method, we determine the first 200 validation images on ImageNet that are classified correctly, and apply the attacks to them by selecting uniformly random attack target classes. Please refer to Appendix D.2 for further details.

Table 2 shows the results. For the targeted ℓ_2 attacks, choosing the value of β from $[0.003162, 0.1]$ provides the improved robustness of DB with the maximum at $\beta = 0.01$. On the other hand, VIB has no improved robustness in all ranges of β. For the targeted ℓ_∞ attacks, DB can reduce the attack success rate even near to 0% (*e.g.* $\beta = 0.003162$ or 0.01). Although VIB decreases the attack success rate to 15.5% at $\beta = 0.1$, VIB already suffers from the performance degradation at $\beta = 0.1$

compared to DB (Figure 3), and increasing β accelerates VIB's degradation even further. Note that the validation accuracy of both VIB and DB are close to zero at $\beta = 0.3162$.

Dimensionality reduction. Figure 3 compares the accuracy of DB and VIB by varying β on the ImageNet validation set. Overall, their accuracies develop similarly with respect to β; while VIB is slightly better in the lower range of β, DB produces better accuracy in the higher range of β. Note that *DB (determ.)* shows the almost identical accuracy plot with *DB*. Importantly, *DB (determ.)* still achieves a reasonable validation accuracy ($\geq 75\%$) using only a few feature dimensions (*e.g.* 8) out of the original 1024 dimensions. This suggests that DB's deterministic compressed representation can greatly reduce the feature dimensionality for inference with only a small trade-off with the performance. It is useful for improving the efficiency of the model after the training is complete. On the other hand, VIB has no such capability. Finally, as Figure 3 shows, the trade-off between the dimensionality reduction and the performance can be controlled by the value of β.

Comparison with feature selection. As the deterministic representation of DB, *DB (determ.)*, provides the dimensionality reduction, we also empirically compare DB with the univariate mutual information-based feature selection method for obtaining the feature space with a reduced dimensionality. In the experiments, the same features provided to DB and VIB are given as input to the feature selection method as well, and for a more straightforward comparison, we let the feature selection method preserve the same number of features as *DB (determ.)*. Refer to Appendix D.3 for further details of the feature selection procedure. Figure 4 shows the classification accuracy of the two methods for the same numbers of features. The results show that while the mutual information-based feature selection method could provide a marginal performance benefit when a larger subset of the pre-trained features is preserved, DB is significantly better at retaining the accuracy with a small number of feature dimensions. For instance, DB achieves the accuracy over 71% even with 4 features, but the accuracy of feature selection method drops from $\approx 68\%$ to $\approx 10\%$ when the number of features is $< 2^6$. Also, we make a comparison of the two methods' adversarial robustness, as well. As Table 3 suggests, the features preserved with the feature selection method show almost no robustness to the targeted ℓ_2 and ℓ_∞ attacks, where every attack success rate is $\geq 97\%$. On the other hand, *DB (determ.)* can reduce the success rate to 20% for the ℓ_2 and to 2% for the ℓ_∞ attacks with 8 features.

6 Conclusion

We presented Drop-Bottleneck as a novel information bottleneck method where compression is done by discretely dropping input features, taking into account each input feature's relevance to the target variable. We then proposed an exploration method based on Drop-Bottleneck, and it showed state-of-the-art performance on multiple noisy and reward-sparse navigation environments from VizDoom and DMLab. The results showed the robustness of Drop-Bottleneck's compressed representation against noise or task-irrelevant information. With experiments on ImageNet, we also showed that Drop-Bottleneck achieves better adversarial robustness compared to VIB and can reduce the feature dimension for inference. In the exploration experiments, we directly fed the noisy observations to the policy, which can be one source of performance degradation in noisy environments. Therefore, applying Drop-Bottleneck to the policy network can improve its generalization further, which will be one interesting future research.

References

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Rana Ali Amjad and Bernhard Claus Geiger. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019b.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Matthew Chalk, Olivier Marre, and Gasper Tkacik. Relevant sparse codes with variational information bottleneck. In *Advances in Neural Information Processing Systems*, pp. 1957–1965, 2016.

Bin Dai, Chen Zhu, and David Wipf. Compressing neural networks using the variational information bottleneck. *arXiv preprint arXiv:1802.10399*, 2018.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 315–323, 2011.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew M Botvinick, Hugo Larochelle, Sergey Levine, and Yoshua Bengio. Infobot: Transfer and exploration via the information bottleneck. *ArXiv*, abs/1901.10902, 2019.

Anirudh Goyal, Yoshua Bengio, Matthew M Botvinick, and Sergey Levine. The variational bandwidth bottleneck: Stochastic evaluation on an information budget. *ArXiv*, abs/2004.11935, 2020.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cynthia H Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *NeurIPS*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2016.

Youngjin Kim, Wontae Nam, Hyunwoo Kim, Ji-Hoon Kim, and Gunhee Kim. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pp. 3379–3388, 2019.

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Artemy Kolchinsky, Brendan D Tracey, and David H Wolpert. Nonlinear information bottleneck. *Entropy*, 21(12):1181, 2019.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database, 2010.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. Invariant representations without adversarial training. In *Advances in Neural Information Processing Systems*, pp. 9084–9093, 2018.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, 2010.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2017.

Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2): e87357, 2014.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

DJ Strouse and David J Schwab. The deterministic information bottleneck. *Neural computation*, 29 (6):1611–1630, 2017.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

A THE SOURCE CODE

We provide our source code in the supplementary zip file, including the anonymized source code for our Drop-Bottleneck exploration method for DMLab (Beattie et al., 2016) tasks. Please refer to `README.md` for the instructions.

B REMOVAL OF TASK-IRRELEVANT INFORMATION AND VALIDITY OF DETERMINISTIC COMPRESSED REPRESENTATION

We experiment Drop-Bottleneck (DB) and Variational Information Bottleneck (VIB) (Alemi et al., 2017) on occluded image classification tasks to show the following:

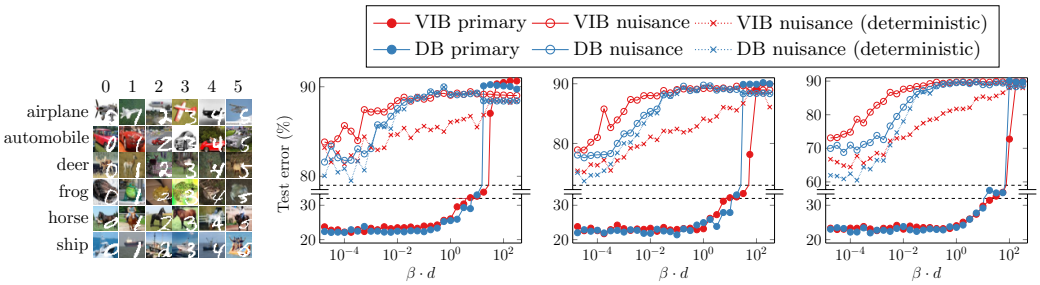
- DB can control the degree of compression (*i.e.* degree of removal of task-irrelevant information) in the same way with VIB as the popular IB method.
- DB's deterministic compressed representation works as a reasonable replacement for its stochastic compressed representation and it maintains the learned indistinguishability better than the attempt of VIB's deterministic compressed representation.

We employ the Occluded CIFAR dataset using the experimental settings from Achille & Soatto (2018). The Occluded CIFAR dataset is created by occluding CIFAR-10 (Krizhevsky, 2009) images with MNIST (LeCun et al., 2010) images as shown in Figure 5a, and each image has two labels of CIFAR and MNIST. We use a modified version of All-CNN-32 (Achille & Soatto, 2018) equipped with an IB method (either of DB or VIB) for the feature extractor whose output dimension is d. Each run consists of two phases. In the first phase, we train the feature extractor with a logistic classifier using both the classification loss for CIFAR and the compression objective of the IB method. Fixing the trained feature extractor, we train a logistic classifier for MNIST in the second phase. We train two different versions of classifiers for each of VIB and DB using stochastic or deterministic compressed representation from the feature extractor. For the deterministic representation of VIB, we use the mode of the output Gaussian distribution.

Figures 5b–5d contain the experimental results with $d = 32$, $d = 64$, and $d = 128$. In the first phase, DB retains only a subset of features that concentrate more on the CIFAR part of the images. Thus, the trained feature extractor preserves less information about the MNIST parts, and the errors of the MNIST classification are high. The first observation is that for both DB and VIB with the original stochastic compressed representation, *nuisance* plots show that increasing β from the minimum value to $\sim 0.1/d$ barely changes the *primary* CIFAR errors but grows the *nuisance* MNIST errors up to $\sim 90\%$ (*i.e.* the maximum error for the 10-way classification). With even larger β, enforcing stronger compression results in the increase of the *primary* errors too, as shown in *primary* plots. This suggests that both DB and VIB provide fine controllability over the removal of task-irrelevant information.

Secondly, if we move our focus to the *nuisance (deterministic)* plots in Figures 5b–5d, which show the test errors on the *nuisance* MNIST classification with the feature extractor's deterministic representation, the results become different between DB and VIB. In DB, the *nuisance (deterministic)* plots follow the *nuisance* plots in the range of β where the compression takes effect (*i.e.* where the *nuisance* errors increase). Moreover, the two plots get closer as β increases. It means that Drop-Bottleneck's deterministic compressed representations maintain the majority of the indistinguishability for the task-irrelevant information learned during the first phase, especially when β is large enough to enforce some degree of the compression. On the other hand, VIB's *nuisance (deterministic)* plots are largely different from the *nuisance* plots; even the *primary* errors rise before the *nuisance (deterministic)* errors reach their maximum values. This shows that employing the mode of VIB's output distribution as its deterministic representation results in loss of the learned indistinguishability.

In summary, we confirm that DB provides controllability over the degree of compression in a similar way as VIB. On the other hand, DB's deterministic representation can be a reasonable replacement for its original stochastic representation in terms of preserving the learned indistinguishability, which is not exhibited by VIB.



(a) Sample images (b) $d = 32$ (c) $d = 64$ (d) $d = 128$

Figure 5: (a) A few samples from Occluded CIFAR dataset (Achille & Soatto, 2018). (b)–(d) Test error plots on the *primary* task (*i.e.* the classification of occluded CIFAR images) and on the *nuisance* tasks (*i.e.* classification of the MNIST digits). For all the three types of tasks, we use the same feature extractor trained for the *primary* task, where its deterministic representation is used only for training and test of the *nuisance (deterministic)* task.

C Ablation Study: Exploration without Drop-Bottleneck

We perform an ablation study to show Drop-Bottleneck (DB)'s ability of dealing with task-irrelevant input information. We examine the performance of the same exploration method as described in Section 4 but without DB; that is, the feature vectors are fully used with no dropping. In order to emphasize the effectiveness of DB for noisy and task-irrelevant information, we conduct experiments with both noisy and original (*i.e.* without explicit noise injection) settings.

Table 4: Comparison of the average episodic sum of rewards in DMLab tasks (over 30 runs), where PPO + Ours (No-Drop-Bottleneck) denotes our exploration method without DB. The original (*i.e.* without explicit noise injection) and three noisy settings are tested: Image Action (IA), Noise (N), Noise Action (NA) and Original (O). The values are measured after 20M (4 action-repeated) time steps, with no seed tuning done. Baseline results for DMLab are cited from Savinov et al. (2019).

Method	DMLab			
	Very Sparse			
	IA	N	NA	O
PPO (Schulman et al., 2017)	6.3	8.7	6.1	8.6
PPO + ICM (Pathak et al., 2017)	4.9	6.0	5.7	11.2
PPO + EC (Savinov et al., 2019)	7.4	13.4	11.3	24.7
PPO + ECO (Savinov et al., 2019)	16.8	26.0	12.5	40.5
PPO + Ours (No-Drop-Bottleneck)	14.9	11.7	10.3	33.0
PPO + Ours (Drop-Bottleneck)	**28.8**	**29.1**	**26.9**	**42.7**
Improvement (%)	93.3	148.7	161.2	29.4

Table 4 shows the results on "Very Sparse" DMLab environments with "Image Action", "Noise", "Noise Action" and "Original" settings. Compared to "Original" setting where observations contain only implicit, inherent noisy information irrelevant to state transitions, DB brings much more significant improvement to exploration methods in "Image Action", "Noise", and "Noise Action" settings, which inject explicit, severe transition-irrelevant information to observations. These results suggest that DB plays an important role handling noisy or task-irrelevant input information.

Table 5: Hyperparameters of PPO (Schulman et al., 2017), PPO + ICM (Pathak et al., 2017), PPO + ECO (Savinov et al., 2019), and PPO + Ours for the VizDoom experiments.

	PPO	PPO + ICM	PPO + ECO	PPO + Ours
PPO				
Learning rate	0.00025	0.00025	0.00025	0.00025
Entropy coefficient	0.01	0.01	0.01	0.01
Task reward scale	5	5	5	5
Exploration method				
Training period and sample size	–	3K	120K	10.8K
# of optimization epochs	–	4	10	4
Intrinsic bonus scale	–	0.01	1	0.001

Table 6: Hyperparameters of PPO (Schulman et al., 2017), PPO + ICM (Pathak et al., 2017), PPO + EC/ECO (Savinov et al., 2019), and PPO + Ours for the DMLab experiments.

	PPO	PPO + ICM	PPO + EC/ECO	PPO + Ours
PPO				
Learning rate	0.00019	0.00025	0.00025	0.00025
Entropy coefficient	0.0011	0.0042	0.0021	0.0011
Task reward scale	1	1	1	1
Exploration method				
Training period and sample size	–	3K	720K (ECO)	21.6K
# of optimization epochs	–	4	10 (ECO)	2
Intrinsic bonus scale	–	0.55	0.030	0.005

D Details of Experiments

We describe additional details of the experiments in Section 5. For all the experiments with DB, each entropy $H(X_i)$ is computed with the binning-based estimation using 32 bins, and the drop probability $\boldsymbol{p}$ is initialized with $p_i = \sigma(p'_i)$ and $p'_i \sim \text{Uniform}(a, b)$ for $a = -2, b = 1$.

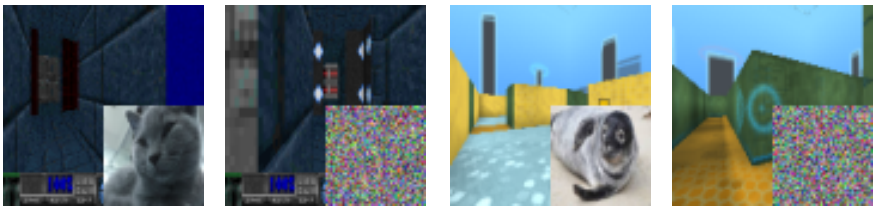
D.1 Details of Exploration Experiments

To supplement Section 5.1, we describe additional details of the VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016) exploration experiments. We collect training samples in a buffer and update $\boldsymbol{p}, T_\psi, f_\phi$ with Equation (14) periodically. We use Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0001 and a batch size of 512. In each optimization epoch, the training samples from the buffer are re-shuffled. For each mini-batch, we optimize the Deep Infomax (Hjelm et al., 2019) discriminator T_ψ with 8 extra epochs with the same samples, to make the Jensen-Shannon mutual information bound tighter. This way of training T_ψ only runs forward and backward passes on T_ψ for the fixed output of the feature extractor f_ϕ, and thus can be done with low computational cost. β is the hyperparameter that determines the relative scales of the compression term and the Deep Infomax Jensen-Shannon mutual information estimator. It is tuned to $\beta = 0.001/128$ for DB and $\beta = 0.0005/128$ for VIB. To make experiments simpler, we normalize our intrinsic rewards with the running mean and standard deviation.

Table 5 and Table 6 report the hyperparameters of the methods for VizDoom and DMLab experiments, respectively. We tune the hyperparameters based on the ones provided by Savinov et al. (2019). Unless specified, we use the same hyperparameters with Savinov et al. (2019).

Under the three noise settings suggested in Savinov et al. (2019), the lower right quadrant of every observation is occupied by a TV screen as follows.

- "Image Action": Every time the agent performs a specific action, it changes the channel of the TV randomly to one of the 30 predefined animal images.
- "Noise": At every observation, a new noise pattern is sampled and shown on the TV screen (independently from the agent's actions).



(a) VizDoom (b) DMLab

Figure 6: Example observations from VizDoom (Kempka et al., 2016) and DMLab (Beattie et al., 2016) environments with "Image Action" (first) and "Noise" (second) settings.

- "Noise Action": Same as "Noise", but the noise pattern only changes when the agent does a specific action.

Figure 6 shows some observation examples from VizDoom and DMLab environments with "Image Action" and "Noise" settings.

D.2 Details of Adversarial Robustness Experiments

For the experiments, each input image is sized as $299 \times 299 \times 3$ and each pixel value is ranged $[-1, 1]$. We perform our adversarial robustness experiments based on the official source code[2] of Carlini & Wagner (2017). For the targeted ℓ_2 attack we increase the number of binary search steps to 20 and use a batch size of 25. Other than the two, the default hyperparameters are used.

D.3 Details of Running Feature Selection Method

The input of the feature selection method is the same as DB's and VIB's: the input features are obtained with a pre-trained model of Inception-ResNet-v2 on ImageNet (Russakovsky et al., 2015). We randomly pick $100k$ samples out of the total 1281167 training samples of ImageNet, and compute the relevance scores for features using those samples by estimating the mutual information between each feature and its label using the entropy estimation with the k-nearest neighbors (Kraskov et al., 2004; Ross, 2014). Given the computed scores, we perform the feature selection by preserving the features with the highest scores.

D.4 Details of Occluded Image Classification Experiments

We use a modified version of All-CNN-32 (Achille & Soatto, 2018). The model architecture for feature dimension d is described in Table 7. Batch normalization (Ioffe & Szegedy, 2015) is applied to Conv layers, and the ReLU (Nair & Hinton, 2010; Glorot et al., 2011) activation is used at every hidden layer. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.001. To ensure the convergence, in each training, the model is trained for 200 epochs with a batch size of 100.

[2] `https://github.com/carlini/nn_robust_attacks`.

Table 7: The network architecture for the occluded image classification experiments.

Input image $32 \times 32 \times 3$		
Feature extractor	Conv [3×3, 96, stride 1] Conv [3×3, 96, stride 1] Conv [3×3, 96, stride 2]	
	Conv [3×3, 192, stride 1] Conv [3×3, 192, stride 1] Conv [3×3, 192, stride 2]	
	FC [d]	FC [$2d$]
	DB [d]	VIB [d]
Classifier	FC [10]	
	softmax	