Seal Your Backdoor with Variational Defense

Supplementary Material

A. Detailed derivation of VIBE E-step

Given a set of trainable parameters θ, our expectation step infers outputs of approximate clean class posterior q for dataset examples as:

$$\frac{1}{N}\ell_{\text{ELBO}}(q|\theta,\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{l^i\sim q(\cdot|\boldsymbol{x}^i,y^i)}\left[\ln p_\theta(y^i|l^i,\boldsymbol{x}^i)+\ln p_\theta(l^i|\boldsymbol{x}^i)-\ln q(l^i|\boldsymbol{x}^i,y^i)\right] \tag{11}$$

$$= \sum_{i=1}^{N}\sum_{l=1}^{K}\underbrace{\frac{1}{N}q(l|\boldsymbol{x}^i,y^i)}_{\mathbf{Q}_{i,l}}\ln[\underbrace{p_\theta(y^i|l,\boldsymbol{x}^i)p_\theta(l|\boldsymbol{x}^i)}_{\mathbf{P}_{i,l}}] - \frac{1}{N}q(l|\boldsymbol{x}^i,y^i)\ln\left[N\frac{1}{N}q(l|\boldsymbol{x}^i,y^i)\right] \tag{12}$$

$$= \sum_{i=1}^{N}\sum_{l=1}^{K}\mathbf{Q}_{i,l}\ln\mathbf{P}_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{K}\mathbf{Q}_{i,l}\ln\mathbf{Q}_{i,l} - \sum_{i=1}^{N}\sum_{l=1}^{K}\mathbf{Q}_{i,l} - \ln N \tag{13}$$

$$= \operatorname{tr}(\mathbf{Q}^\top\ln\mathbf{P}) + \mathbb{H}(\mathbf{Q}) + 1 - \ln N \tag{14}$$

$$\geq \operatorname{tr}(\mathbf{Q}^\top\ln\mathbf{P}) + \frac{1}{\lambda}\mathbb{H}(\mathbf{Q}) + \underbrace{1-\ln N}_{\text{const.}} \tag{15}$$

Here, we defined the discrete entropy of a transport matrix $\mathbb{H}(\mathbf{Q})$ as $\mathbb{H}(\mathbf{Q}) := -\sum_{i,j}\mathbf{Q}_{i,j}(\ln\mathbf{Q}_{i,j}-1) = \operatorname{tr}(\mathbf{Q}^\top(1-\ln\mathbf{Q}))$, following [72]. The inequality holds since $\lambda > 1$ while $1-\ln N$ is a constant dependent on training dataset size.

We observe that by the definition of $\mathbf{Q}$ each matrix row $\mathbf{Q}_{i,:}$ sums to $1/N$. Further assuming some prior over classes $\boldsymbol{\pi}$, the set of all possible solutions $\mathbf{Q}$ of the objective (4) forms a polytope:

$$\mathcal{Q}[\boldsymbol{\pi}] = \{\mathbf{Q}\in\mathbb{R}_+^{N\times K}\,|\,\mathbf{Q}^\top\mathbf{1}_N = \boldsymbol{\pi},\ \mathbf{Q}\mathbf{1}_K = \frac{1}{N}\mathbf{1}_N\}. \tag{16}$$

Solving the objective (15) on polytope $\mathcal{Q}[\boldsymbol{\pi}]$ corresponds to an entropy-regularized optimal transport problem:

$$\max_{\mathbf{Q}\in\mathcal{Q}[\boldsymbol{\pi}]}\operatorname{tr}(\mathbf{Q}^\top\ln\mathbf{P}) + \frac{1}{\lambda}\mathbb{H}(\mathbf{Q}) - \ln N + 1 = \min_{\mathbf{Q}\in\mathcal{Q}[\boldsymbol{\pi}]} -\operatorname{tr}(\mathbf{Q}^\top\ln\mathbf{P}) - \frac{1}{\lambda}\mathbb{H}(\mathbf{Q}) \tag{17}$$

Solving entropy-regularized optimal transport problems can be conveniently done by the Sinkhorn-Knopp matrix scaling algorithm [15, 72]. For completeness, we next revisit the matrix scaling algorithm.

B. Sinkhorn-Knopp matrix scaling algorithm for VIBE objective

We revisit the solution to the entropy-regularized optimal transport problem via Sinkhorn-Knopp matrix scaling algorithm [72] in the case of cost matrix $\mathbf{M} = -\ln\mathbf{P}$ and polytope $\mathcal{Q}[\boldsymbol{\pi}]$. Let $\boldsymbol{a}\in\mathbb{R}^N$ and $\boldsymbol{b}\in\mathbb{R}^K$ be two dual variables that correspond to the polytope constraints (5). We can write the corresponding Lagrangian as:

$$L(\mathbf{Q},\boldsymbol{a},\boldsymbol{b}) = -\operatorname{tr}(\mathbf{Q}^\top\ln\mathbf{P}) - \frac{1}{\lambda}\mathbb{H}(\mathbf{Q}) + \boldsymbol{a}^\top(\mathbf{Q}\mathbf{1}_K - \frac{1}{N}\mathbf{1}_N) + \boldsymbol{b}^\top(\mathbf{Q}^\top\mathbf{1}_N - \boldsymbol{\pi}). \tag{18}$$

Rewriting $L(\mathbf{Q},\boldsymbol{a},\boldsymbol{b})$ using matrix trace operators and following trace rules gives us:

$$L(\mathbf{Q},\boldsymbol{a},\boldsymbol{b}) = -\operatorname{tr}[(\ln\mathbf{P})^\top\mathbf{Q}] + \frac{1}{\lambda}\operatorname{tr}[(\ln\mathbf{Q}-1)^\top\mathbf{Q}] + \operatorname{tr}[(\boldsymbol{a}\mathbf{1}_K^\top)^\top\mathbf{Q}] - \operatorname{tr}[\boldsymbol{a}^\top\frac{1}{N}\mathbf{1}_N] + \operatorname{tr}[(\mathbf{1}_N\boldsymbol{b}^\top)^\top\mathbf{Q}] - \operatorname{tr}[\boldsymbol{b}\boldsymbol{\pi}^\top]. \tag{19}$$

Aggregating all elements with $\mathbf{Q}$ gives us:

$$L(\mathbf{Q},\boldsymbol{a},\boldsymbol{b}) = \operatorname{tr}\left[\left(-\ln\mathbf{P} + \frac{1}{\lambda}(\ln\mathbf{Q}-1) + \boldsymbol{a}\mathbf{1}_K^\top + \mathbf{1}_N\boldsymbol{b}^\top\right)^\top\mathbf{Q}\right] - \operatorname{tr}[\boldsymbol{a}^\top\frac{1}{N}\mathbf{1}_N)] - \operatorname{tr}[\boldsymbol{b}\boldsymbol{\pi}^\top]. \tag{20}$$

Fixing $\mathbf{a}$ and $\mathbf{b}$ and expressing differential dL in terms of $d\mathbf{Q}$ equals:

$$dL = \operatorname{tr}\left[\left(-\ln \mathbf{P} + \frac{1}{\lambda}\ln \mathbf{Q} + \boldsymbol{a}\mathbf{1}_K^\top + \mathbf{1}_N \boldsymbol{b}^\top\right)^\top d\mathbf{Q}\right]. \tag{21}$$

Recall that $dL = \langle \frac{\partial L}{\partial \mathbf{Q}}, d\mathbf{Q}\rangle = \operatorname{tr}\left[\left(\frac{\partial L}{\partial \mathbf{Q}}\right)^\top d\mathbf{Q}\right]$, so we have precomputed the $\frac{\partial L}{\partial \mathbf{Q}}$ in (21). By further setting $\frac{\partial L}{\partial \mathbf{Q}} = 0$ we get:

$$\mathbf{Q} = \exp(-\lambda \cdot \boldsymbol{a}\mathbf{1}_K^\top) \odot \mathbf{P}^\lambda \odot \exp(-\lambda \cdot \mathbf{1}_N \boldsymbol{b}^\top) = \operatorname{diag}(\boldsymbol{u})\mathbf{P}^\lambda \operatorname{diag}(\boldsymbol{v}). \tag{22}$$

Here, $\boldsymbol{u} = \exp(-\lambda \cdot \boldsymbol{a})$ and $\boldsymbol{v} = \exp(-\lambda \cdot \boldsymbol{b})$ are two vectors with strictly positive elements and $\odot$ is elementwise product. Since every solution $\mathbf{Q} \in \mathcal{Q}[\boldsymbol{\pi}]$, the following equalities hold:

$$\operatorname{diag}(\boldsymbol{u})\mathbf{P}^\lambda \operatorname{diag}(\boldsymbol{v})\mathbf{1}_K = \frac{1}{N}\mathbf{1}_N \quad \text{and} \quad \operatorname{diag}(\boldsymbol{v})(\mathbf{P}^\lambda)^\top \operatorname{diag}(\boldsymbol{u})\mathbf{1}_N = \boldsymbol{\pi}. \tag{23}$$

The two equalities can be further rewritten as:

$$\boldsymbol{u} \odot (\mathbf{P}^\lambda \boldsymbol{v}) = \frac{1}{N}\mathbf{1}_N \quad \text{and} \quad \boldsymbol{v} \odot ((\mathbf{P}^\lambda)^\top \boldsymbol{u}) = \boldsymbol{\pi}. \tag{24}$$

By finding $\boldsymbol{u}$ and $\boldsymbol{v}$ that satisfy the above conditions we will effectively recover the solution $\mathbf{Q}$ [64], as indicated by the equation (22). Thus, we resort to iterative updates of the two vectors, following Sinkhorn's algorithm:

$$\boldsymbol{u}_{t+1} := \frac{\mathbf{1}_N}{N \cdot \mathbf{P}^\lambda \boldsymbol{v}_t} \quad \text{and} \quad \boldsymbol{v}_{t+1} := \frac{\boldsymbol{\pi}}{(\mathbf{P}^\lambda)^\top \boldsymbol{u}_t}. \tag{25}$$

Running the matrix scaling algorithm on modern GPU hardware makes our approach feasible even for large-scale datasets.

C. Detailed derivation of VIBE M-step

Given approximate clean label posterior for dataset examples, we turn to optimization of parameters θ. We rewrite the maximization of objective ℓ_{ELBO} as:

$$\max_\theta \ \ell_{\text{ELBO}}(\theta|q,\mathcal{D}) = \min_\theta \ -\sum_{i=1}^N \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i, y^i)}\left[\ln p_\theta(y^i|l^i,\boldsymbol{x}^i) + \ln p_\theta(l^i|\boldsymbol{x}^i) - \ln q(l^i|\boldsymbol{x}^i,y^i)\right] \tag{26}$$

$$= \min_\theta \ \sum_{i=1}^N \sum_{l=1}^K q(l|\boldsymbol{x}^i,y^i)\left[-\ln p_\theta(y^i|l,\boldsymbol{x}^i) - \ln p_\theta(l|\boldsymbol{x}^i) + \ln q(l|\boldsymbol{x}^i,y^i)\right] \tag{27}$$

$$= \min_\theta \ \sum_{i=1}^N \sum_{l=1}^K -q(l|\boldsymbol{x}^i,y^i)\ln p_\theta(l|\boldsymbol{x}^i) - \sum_{l=1}^K q(l|\boldsymbol{x}^i,y^i)\ln p_\theta(y^i|l,\boldsymbol{x}^i) - \mathbb{H}[q] \tag{28}$$

$$= \min_\theta \ \sum_{i=1}^N \operatorname{CE}[q(l|\boldsymbol{x}^i,y^i)\,||\,p_\theta(l|\boldsymbol{x}^i)] - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i,y^i)}\left[\ln p_\theta(y^i|l^i,\boldsymbol{x}^i)\right] - \mathbb{H}[q] \tag{29}$$

$$\cong \min_\theta \ \sum_{i=1}^N \operatorname{CE}[q(l|\boldsymbol{x}^i,y^i)\,||\,p_\theta(l|\boldsymbol{x}^i)] - \mathbb{E}_{l^i \sim q(\cdot|\boldsymbol{x}^i,y^i)}\left[\ln p_\theta(y^i|l^i,\boldsymbol{x}^i)\right] \tag{30}$$

Here, $\mathbb{H}$ is the entropy term independent of the parameters and thus can be ignored.

D. VIBE training algorithm

Algorithm 1 shows the pseudocode for VIBE training. Prior to running the algorithm, we apply a preprocessing step that removes off-manifold training data, as described in Section 3.4.

Algorithm 1 EM algorithm for training VIBE

Require: Dataset $\mathcal{D}$, E-step period T, total number of iterations N_{iters}, learning rate ϵ, prior temperature c, entropy regularization coefficient λ
Ensure: Backdoor resilient classifier

$\theta \leftarrow$ initialization($\mathcal{D}$) ▷ Initilization of trainable parameters.
$\mathcal{D} \leftarrow$ remove off-manifold data from the original dataset $\mathcal{D}$ ▷ Data preprocessing as described in Alg. 2
$\mathbf{Q} \leftarrow$ SK-algorithm($-\ln \mathbf{P}, \boldsymbol{\pi}, \lambda$) that solves OT problem (6)
for j in $\{0, \ldots, N_{\text{iters}}\}$ **do**
  if $j \mod T = 0$ **then** ▷ Perform E-step every T iterations.
    $\mathbf{P} \leftarrow$ compute probabilities for $\mathcal{D}$ with parameters θ
    $\mathbf{Q} \leftarrow$ SK-algorithm($-\ln \mathbf{P}, \boldsymbol{\pi}, \lambda$) that solves OT problem (6)
  end if
  $\mathbf{X}, \boldsymbol{y} \leftarrow$ sample minibatch from $\mathcal{D}$ ▷ Perform M-step.
  $\mathcal{L} \leftarrow$ evaluate (7) for $(\mathbf{X}, \boldsymbol{y})$ and $\mathbf{Q}$
  $\theta \leftarrow \text{SGD}(\mathcal{L}, \epsilon)$
end for

E. Parameterizing VIBE posteriors

E.1. Clean class posterior

We define likelihood of the encoded input $\boldsymbol{v}^i = g_\theta(\boldsymbol{x}^i)$ conditioned on the clean class l^i as a von Mises-Fisher distribution [5]:

$$p_\theta(\boldsymbol{v}^i | l^i) := C_d(\kappa) \exp(\kappa \boldsymbol{\mu}_{l^i}^\top \boldsymbol{v}^i). \tag{31}$$

The vector $\boldsymbol{\mu}_{l^i} \in S^{d-1}$ sets the mean direction, the hyper-parameter κ controls the distribution spread, while $C_d(\kappa)$ is a normalization constant [5]. We set $p_\theta(\boldsymbol{v}^i | \boldsymbol{x}^i) := \delta(\boldsymbol{v}^i - g_{\theta_{\text{E}}}(\boldsymbol{x}^i))$ since feature extractor $g_{\theta_{\text{E}}}$ encodes $\boldsymbol{x}^i$ exactly into $\boldsymbol{v}^i$. By assuming a prior over clean classes $p_\pi(l^i)$, the clean class posterior can now be recovered with the Bayes rule:

$$p_\theta(l^i | \boldsymbol{x}^i) = \int p_\theta(l^i | \boldsymbol{v}^i) p_\theta(\boldsymbol{v}^i | \boldsymbol{x}^i) \, d\boldsymbol{v}^i = \int p_\theta(l^i | \boldsymbol{v}^i) \, \delta(\boldsymbol{v}^i - g_{\theta_{\text{E}}}(\boldsymbol{x}^i)) \, d\boldsymbol{v}^i \tag{32}$$

$$= p_\theta(l^i | \boldsymbol{v}^i = g_{\theta_{\text{E}}}(\boldsymbol{x}^i)) \tag{33}$$

$$= \frac{p_\theta(\boldsymbol{v}^i | l^i) p_\pi(l^i)}{\sum_{l'} p_\theta(\boldsymbol{v}^i | l') p_\pi(l')} = \frac{p_\theta(\boldsymbol{v}^i | l^i) \pi_{l^i}}{\sum_{l'} p_\theta(\boldsymbol{v}^i | l') \pi_{l'}} = \frac{\exp(\kappa \, {\boldsymbol{v}^i}^\top \boldsymbol{\mu}_{l^i} + \ln \pi_{l^i})}{\sum_{l'} \exp(\kappa \, {\boldsymbol{v}^i}^\top \boldsymbol{\mu}_{l'} + \ln \pi_{l'})}. \tag{34}$$

The mixing coefficients $\boldsymbol{\pi}$ are induced by a learnable prior over clean classes, *i.e.* a categorical distribution $p_\pi(l^i) := \pi_{l^i}$. In practice, we set $\boldsymbol{\pi} = \sigma(c \cdot \theta_\pi)$, where σ is softmax activation, c is a hyperparmeter, and $\theta_\pi \in \mathbb{R}^d$ are learnable parameters. This reparametrization ensures both gradient updates towards θ_π due to closed form derivation of $d\boldsymbol{\pi}/d\theta_\pi$ as well as $\sum_{l=1}^K \pi_l = 1$.

E.2. Corrupted class posterior

We define the corrupted class posterior $p_\theta(y^i | l^i, \boldsymbol{x}^i)$ as cosine similarity between the corrupted class prototypes $\theta_y = \{\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_K\}$ and output of function h that process the encoded input $\boldsymbol{v}^i$ and the clean label prototype $\boldsymbol{\mu}_{l^i}$:

$$p_\theta(y^i | l^i, \boldsymbol{x}^i) := \frac{\exp(\nu \cdot {\boldsymbol{\eta}_{y^i}}^\top h(\boldsymbol{\mu}_{l^i}, \boldsymbol{v}^i))}{\sum_{y'} \exp(\nu \cdot {\boldsymbol{\eta}_{y'}}^\top h(\boldsymbol{\mu}_{l^i}, \boldsymbol{v}^i))}. \tag{35}$$

Here, ν is a scalar hyper-parameter, while details on h are deferred to implementation details.

We can approximate $p_\theta(y^i | l^i, \boldsymbol{x}^i)$ by removing the dependence on inputs. The approximate corrupted class posterior equals to:

$$p_\theta(y^i | l^i) := \frac{\exp(\nu \cdot {\boldsymbol{\eta}_{y^i}}^\top \boldsymbol{\mu}_{l^i})}{\sum_{y'} \exp(\nu \cdot {\boldsymbol{\eta}_{y'}}^\top \boldsymbol{\mu}_{l^i})}. \tag{36}$$

The approximated posterior (36) hardens the optimization of ℓ_{ELBO} but enables seamless recovery of data poisoning rules.

F. Preprocessing via manifold learning

The key insight behind our approach is that examples poisoned with a clean-label attack move away from the data manifold in the feature space of a self-supervised model pretrained on the poisoned dataset instance. This behavior arises from the necessity for clean-label attacks to introduce substantial perturbations to poisoned images in order to be effective. Consequently, we can defend against them by capturing the data manifold and removing the furthest community. Algorithm 2 details our preprocessing strategy.

Algorithm 2 VIBE preprocessing strategy

Require: Dataset $\mathcal{D}$, pretrained feature extractor g_θ, threshold δ, number of nearest neighbors k, number of classes K.
Ensure: Postprocessed dataset $\mathcal{D}$

$\mathbf{F} \leftarrow$ compute features for dataset $\mathcal{D}$ with g_θ
$A_k \leftarrow$ construct k-NN graph from $\mathbf{F}$
$\mathcal{C} \leftarrow$ discover K+1 communities in A_k with Leiden algorithm
for i in $\{0, \ldots, K\}$ **do**
    $\boldsymbol{d}[i] \leftarrow$ average pairwise distance between the i-th community and other communities $\mathcal{C}/i$
end for
$c \leftarrow \arg\max_i \boldsymbol{d}[i]$ ▷ Select the index of most distant community.
if $\boldsymbol{d}[c] > \delta$ **then** ▷ Check if distant community is far away.
    $\mathcal{D} \leftarrow$ remove examples belonging to c-th community from the dataset
else
    Keep all examples in $\mathcal{D}$
end if

Figure 7 shows the implications of the preprocessing strategy on clean-label attacks LC [88], SIG [6] and CLBA [107], as well as poisoned-label attack BadNets [24]. In the presence of clean-label attacks, the captured off-manifold community corresponds to poisoned examples that are later safely removed (rows 2-4). In the absence of clean-label attacks the captured outlier community corresponds to a fistful of outlier samples that are still close enough to the data manifold and thus are retained (row 1). Note that the visualized distances may be distorted due to two-dimensional UMAP [60] plots of high-dimensional space.

G. Details on experimental setup

This section outlines details of the considered baseline attacks and defenses.

G.1. Details on backdoor attacks

BadNets. We follow [20, 34] by adopting the same 2×2 trigger pattern for CIFAR-10 and CIFAR-100 and Apple logo for ImageNet.
Blend. Following original work [12] and [20, 34], we use use the *Hello-kitty* trigger pattern on CIFAR datasets and random noise pattern on ImageNet datasets. Blending ratio is set to 0.1.
WaNet. We follow the setup in [20, 34] to generate trigger patterns using warping operations. We set k to 4 on CIFAR datasets and k to 224 on ImageNet datasets.
Frequency. We rely on BackdoorBench[3] to reproduce Frequency attack. All hyper-parameters are set as in [91].
Adap-patch & Adap-blend. We follow the attack setup from [73].
Label Consistent. We use PGD [59] to generate adversarial perturbations within L^∞ ball. Maximum magnitude η is set to 16, step size to 1.5 and perturbation steps to 30. Trigger pattern is 3×3 grid pattern in each corner of the image.
UBA. We adopt the attack setup as in [80]. In the Patch and Blend versions, we poison 2000 and 8000 examples following the original paper.

G.2. Details on baseline defenses

ABL. To reproduce ABL, we use BackdoorBox[4]. We found ABL to be very sensitive to its main hyper-parameter γ. Therefore, we conduct the grid search to find the best $\gamma \in \{0, 0.1, 0.2, 0.5\}$ yielding lowest ASR against every attack. We note

[3] https://github.com/SCLBD/BackdoorBench
[4] https://github.com/THUYimingLi/BackdoorBox

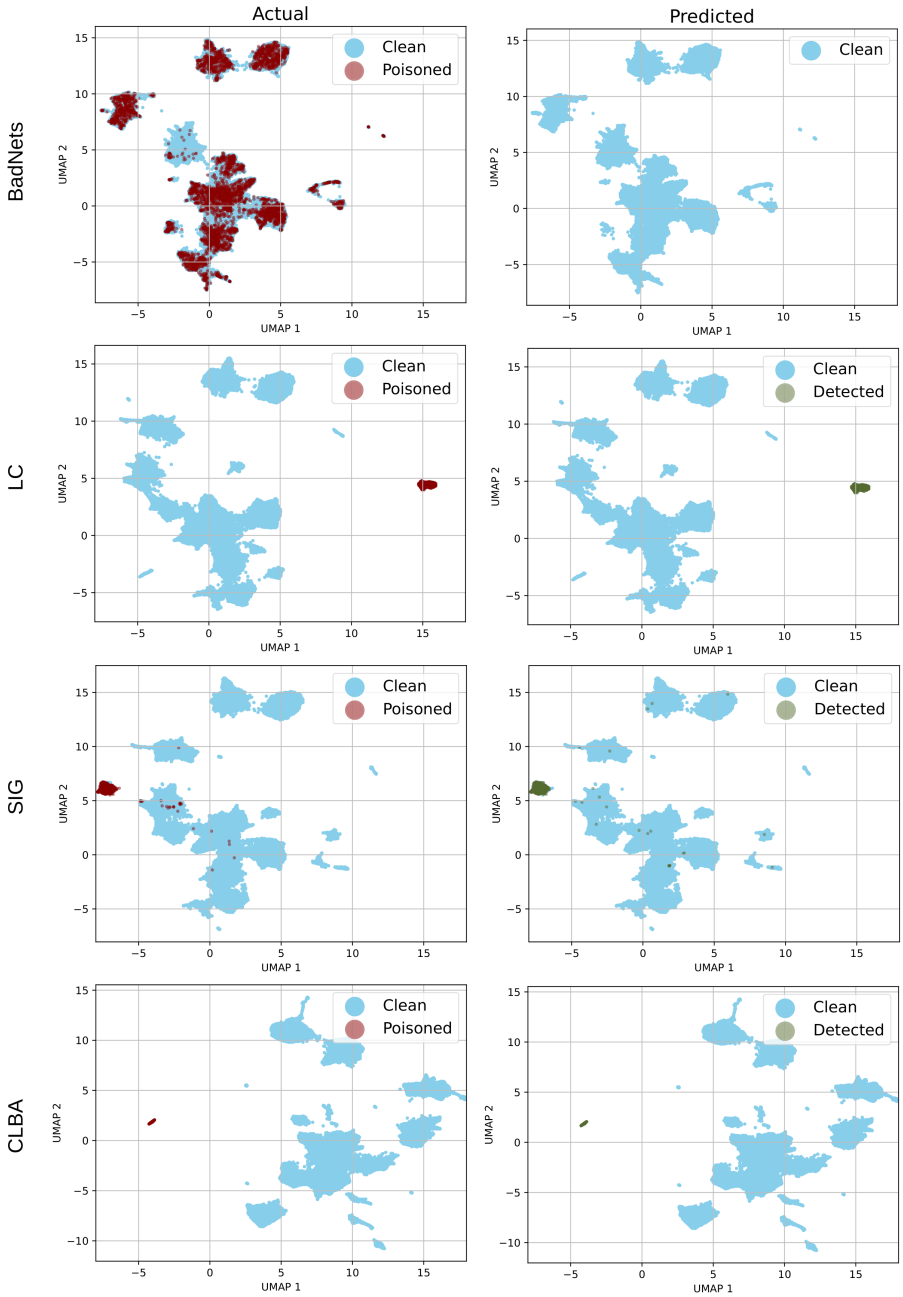

Figure 7. Preprocessing strategy successfully removes poisoned data in all considered clean-label attacks.

that this assumption might be overoptimistic in practice. Following original work [50], we poison the model for 20 epochs, followed by 70 epochs of fine-tuning. Lastly, we unlearn the backdoor for 5 epochs.

DBD. We refer to the official implementation[5] to reproduce DBD and use all configurations as introduces in [34].

CBD. We use the official code implementation[6] and all configurations from [111].

[5] https://github.com/SCLBD/DBD

[6] https://github.com/zaixizhang/CBD

ASD. We rely on official implementation[7] and configurations from original paper [20].
VAB. We rely on official code implementation[8] and follow configurations from original work [115].

H. VIBE implementation details

We use ResNet-18 [29] feature extractor with self-supervised initialization [18] on the poisoned dataset of interest. We train VIBE end-to-end for $30k$ iterations using the proposed EM algorithm. In every iteration, we perform the M-step using SGD with learning rate 10^{-3} and batch size 256. We perform a dataset-wide E-step every $T = 1$k iterations by solving entropy-regularized optimal transport, with $\lambda = 25$, as validated in early experiments. Hyper-parameters ν and κ are set to 10 for datasets with number of classes $K \leq 30$ and to 20 otherwise. We fix $k = 50$, $\delta = 0.275$, and $c = 0.02$ with early validation experiments. Experiments with frozen foundation model involve ViT-G/14 [17] pretrained with DINOv2 [68]. These experiments fix the poisoned class prototypes ψ as means over poisoned labels. Thus, we optimize clean prototypes ϕ for 15k iterations using SGD with constant learning rate 10^{-2} and batch size 1024. We conduct E-step every $T = 500$ iterations. In both setups, we fix $h(\boldsymbol{\mu}, \boldsymbol{v}) = \boldsymbol{\mu} + \boldsymbol{v}$, yet other choices of h may also yield competitive performance. All experiments were conducted to maximize GPU utilization. We measured maximal memory requirements with *torch.cuda.max_memory_allocated*.

I. Extended results

I.1. Extended results for ImageNet-1k dataset

Figure 8 illustrates the difference between ACC and ASR for both VIBE-SS and the DBD baseline. Both methods are built upon a frozen ResNet-50 pretrained on poisoned instances of the ImageNet-1k. VIBE-SS successfully defends against a variety of attacks in this setup, achieving both high ACC and high robustness.

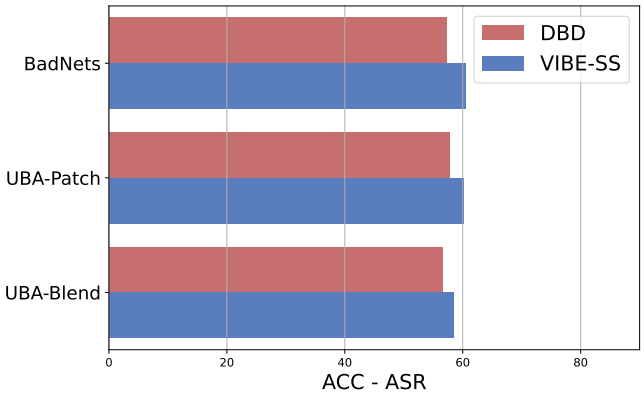

Figure 8. VIBE performance when combined with ResNet-50 pretrained on poisoned instances of the ImageNet-1k dataset.

I.2. Attacks on self-supervision

Relevant backdoor attacks targeting self-supervised learning [46, 79] operate by injecting carefully designed triggers that poison the embedding space of a model pretrained according to contrastive self-supervised objectives. In this setup, attack is successful if the backdoor persists even after model fine-tuning or linear probing on a small subset of clean data. While this setup slightly differs from the one studied in our main paper, VIBE again exhibits competitive performance. Additionally, robust self-supervised pre-training objectives already exist [7, 27]. Modular VIBE design allows seamless integration of such robust pre-training.

Table 6 compares VIBE performance with the most relevant baseline DBD on the CIFAR-10 dataset poisoned with CTRL [46]. In this experiment, we rely on SimCLR [10] pre-training objective to stay consistent with the original setup [46]. The first row of shows the performance of VIBE and DBD baseline against the CTRL attack [46] after the standard SimCLR pretraining. VIBE performance surpasses DBD both in terms of accuracy and ASR. Since robust pre-training methods exist, we validate VIBE and DBD performance atop one such method MIMIC [27]. In this case VIBE becomes even more resilient,

[7] https://github.com/KuofengGao/ASD
[8] https://github.com/Zixuan-Zhu/VaB

attaining ASR of only 4%. In contrast, DBD suffers from a loss of generalization power on the clean data. To understand the cause of this performance drop, we conducted a thorough hyperparameter analysis. We concluded that this behavior stems from the loss of labels due to the filtering strategy characteristic for DBD. Altogether, we find robust self-supervised objectives a promising research direction that can be easily ported into the VIBE framework.

Defense →	LogReg			DBD			VIBE-SS		
Self-sup objective ↓	ACC	ASR	ACC - ASR	ACC	ASR	ACC - ASR	ACC	ASR	ACC - ASR
SimCLR	81.0	64.3	16.7	51.4	30.6	20.8	84.5	26.8	**57.7**
SimCLR + MIMIC	61.9	7.7	54.2	49.8	0.0	49.8	71.6	4.2	**67.4**

Table 6. Defending against the CTRL attack targeting self-supervision. Higher difference between ACC and ASR indicates better performance.

Note that we found the attacks on self-supervision to be fragile in practice. Consistent to the findings in [46], we were unable to reproduce the SSL attack [79]. Furthermore, the effectiveness of the CTRL attack [46] was highly sensitive to how the poisoned data was stored. Specifically, saving the data as `float32` preserved the attack, whereas the standard data storage format `uint8` nullified the poisoning effect.

I.3. Adaptive attack on VIBE

Given that the attacker has knowledge about how our defense operates, they may attempt to construct an attack which bypasses our defense mechanism. Such scenario is regarded as the adaptive attack. Since our defense jumpstarts the optimization process with self-supervised initialization, one adaptive attack would be to construct a trigger such that the self-supervised representations of poisoned examples resemble those of target class examples. Concretely, we optimize the trigger $\boldsymbol{t}$ by minimizing the distance between the representations of poisoned examples and the target class centroid. Given the original dataset $\mathcal{D}_{\text{raw}}$, let $\mathcal{D}_{\text{P}} \in \mathcal{D}$ be the poisoned subset and $\mathcal{D}_{\text{T}} = \{(\boldsymbol{x}, y) : (\boldsymbol{x}, y) \in \mathcal{D}, y = y_T\}$ be the target class subset.

$$\arg\min_{\boldsymbol{t}} \frac{1}{|\mathcal{D}_{\text{P}}|} \sum_{i}^{|\mathcal{D}_{\text{P}}|} \left|\left| \mathcal{B}(g(\boldsymbol{x}_i), \boldsymbol{t}) - \frac{1}{|\mathcal{D}_{\text{T}}|} \sum_{j}^{|\mathcal{D}_{\text{T}}|} g(\boldsymbol{x}_j) \right|\right|_2 \tag{37}$$

$$\text{s.t.} \quad ||\boldsymbol{t}||_\infty \leq \epsilon \tag{38}$$

Settings. We conduct the attack on the CIFAR-10 dataset and set the trigger size to be the same of the original image. $\mathcal{B}(\cdot, \boldsymbol{x})$ is a blending function with factor 0.5. We optimize for 100 iterations with SGD and base learning rate set to 0.1. Learning rate is decayed with 0.1 every 40 iterations and ϵ is set to $\frac{32}{255}$ to conceal the trigger. Poisoned rate is 10%.
Results. For model trained with standard cross-entropy loss, the proposed adaptive attack results in ASR of 99.2% and clean accuracy of 91.2%. On contrary, VIBE is completely robust against such attack. It has no backdoor injected (ASR=0.8%) and generalizes even better on the clean data (ACC=94.6%). A more comprehensive analysis indicates that VIBE corrects the labels of poisoned examples into their actual classes.

I.4. Extended results on clean-label attacks

Table 7 empirically shows that the proposed preprocessing strategy improves VIBE performance for the most relevant clean-label attacks. In the case of the LC attack, VIBE offers significant robustness even without the preprocessing step. Furthermore, the proposed preprocessing does not affect the overall dataset size when the clean-label attacks are absent. Thus, we attain similar performance on the poisoned-label BadNets attack.

Attack →	BadNets		LC		SIG		CLBA	
Preprocess ↓	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
✗	94.5	0.4	94.6	14.0	94.2	45.0	94.6	89.3
✓	94.4	0.4	94.3	6.0	92.7	14.7	94.1	14.1

Table 7. VIBE performance with and without our preprocessing strategy.

Table 8 compares the proposed method against other baseline defenses on the three clean-label attacks. On average, VIBE outperforms all considered baselines. Furthermore, all baselines have some failure modes, such as the CLBA attack. On the contrary, VIBE does not completely fail against any of the three clean-label attacks.

Defense →	No Defense		ABL		DBD		CBD		ASD		VIBE-SS	
Attack ↓	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
LC	94.9	99.9	86.6	1.3	89.7	0.0	91.3	24.7	93.1	0.9	93.3	6.0
SIG	92.3	98.0	81.6	0.1	91.6	100	92.5	1.1	93.4	1.0	92.7	14.7
CLBA	93.2	80.0	84.6	93.4	91.5	87.6	90.4	37.8	93.2	98.0	94.1	14.1
Average	94.1	90.0	84.1	31.6	90.9	62.5	91.9	12.9	93.2	33.3	**93.3**	**11.4**

Table 8. Extended results for clean-label attacks.

Additionally, we test VIBE's resilience against targeted poisoning attacks [21, 81] which aim to misclassify one specific test example. Although the general goal behind these attacks slightly diverges from standard backdoor attacks, the poisoning setup is the same. The attacker aims to inject malicious behaviour into the model by poisoning the training data. Still, VIBE provides complete robustness against Witches Brew [21], a representative of targeted poisoning attacks, by reducing the ASR from 50% to 0%.

I.5. Comparison with post-training defenses

Unlike the training-time defenses such as VIBE, which operate solely on the poisoned dataset, post-training defenses assume access to the poisoned model and additional clean data. Still, the outcome of both categories of defenses is a robust model. VIBE outperforms NAD [49] and ANP [98], representatives of post-training defenses, as shown in the Table 9.

Defense →	NAD		ANP		VIBE-SS	
Data ↓	ACC	ASR	ACC	ASR	ACC	ASR
CIFAR-10	82.9	6.1	90.9	2.1	**94.1**	**1.7**
ImageNet-30	91.1	0.5	-	-	**96.9**	**0.1**

Table 9. Experimental comparison with NAD [49] and ANP [98]. The numbers are averaged over 4 attacks on CIFAR-10 and 3 attacks on ImageNet-30.

I.6. Comparison with detection defenses

Detection defenses focus on identifying poisoned examples. However, retraining on data filtered by these methods may still result in poisoned models, as noted in [69]. Although not being the main focus of our work, VIBE can be used as a poisoned sample detector. The psuedo-labelling in E-step combined with our pre-processing strategy employed to detect clean-label data achieves performance on par with state-of-the-art detection methods SCP [26], BSU [69] and PSBD [47], as demonstrated in Table 10.

Attack ↓	SCP	BSU	PSBD	VIBE-SS
BadNets	.83/**1.0**/.30	.95/**1.0**/.17	**.99**/**1.0**/.10	**.99**/.99/**.01**
Blend	.50/.13/.18	.95/.99/.10	.96/**1.0**/.14	**.98**/.96/**.01**
WaNet	.73/.90/.28	.93/.99/.10	**.99**/**1.0**/.14	**.99**/.98/**.00**
LC	.93/.94/.18	.95/.99/.14	**1.0**/.99/.13	.99/**1.0**/**.02**

Table 10. Experimental comparison with detection defenses. All measurements are in AUROC/TPR/FPR format.

I.7. Different architectures of the feature extractor

Being agnostic to the backbone choice, VIBE can be built atop different model architectures. Table 11 showcases robust performance when VGG-11 [84] is used as backbone.

Defense →	No defense		ASD		VIBE-SS	
Attack ↓	ACC	ASR	ACC	ASR	ACC	ASR
BadNets	91.0	99.9	90.4	3.7	**91.3**	**1.1**
Blend	90.6	98.4	87.5	2.4	**91.3**	**2.2**

Table 11. VIBE-SS performance with VGG-11 as backbone on CIFAR-10 dataset.

J. Hyper-parameter sensitivity

This section analyzes VIBE performance for different values of hyper-parameters. All experiments are conducted on the CIFAR-10 dataset poisoned with BadNets attack. VIBE attains similar performance for different hyper-parameter values.

T	ACC	ASR
500	94.4	0.5
1000	94.4	0.4
2000	94.4	0.5

Table 12. VIBE performance for different values of E-step period T.

lr	ACC	ASR
10^{-2}	92.9	0.9
$\mathbf{10^{-3}}$	94.4	0.4
10^{-4}	91.0	0.6

Table 13. VIBE performance for different values of the learning rate.

δ	ACC	ASR
0.250	94.4	0.5
0.275	94.4	0.4
0.300	94.3	0.5

Table 14. VIBE performance for different distance thresholds δ.

c	ACC	ASR
0.05	94.4	0.6
0.02	94.4	0.4
0.01	94.4	0.5

Table 15. VIBE performance for different values of the temperature c used for prior $\boldsymbol{\pi} = \sigma(c \cdot \theta_\pi)$.

κ	ACC	ASR
20	94.5	0.3
10	94.4	0.4
2	92.7	0.8

Table 16. VIBE performance for different values of temperature κ used in $p_{\phi,\theta}(l|\boldsymbol{x})$.

ν	ACC	ASR
20	94.4	0.5
10	94.4	0.4
2	93.3	0.6

Table 17. VIBE performance for different values of ν used in $p_{\phi,\psi}(y|l)$.

λ	ACC	ASR
10	94.4	0.5
25	94.4	0.4
50	94.5	0.5

Table 18. VIBE performance for different entropy regularization hyperparameters λ.

K. Qualitative results

Figure 9 shows the VIBE latent space after the training. The learned clean class prototypes are denoted with squares while the poisoned class prototypes are denoted with triangles. Poisoned class prototype (blue triangle) of the target class is located in the center of the data manifold because the target class contains examples from all other classes in *all-to-one* poisoning. Contrary, the clean class prototypes learned by VIBE are adequately assigned across the clusters.

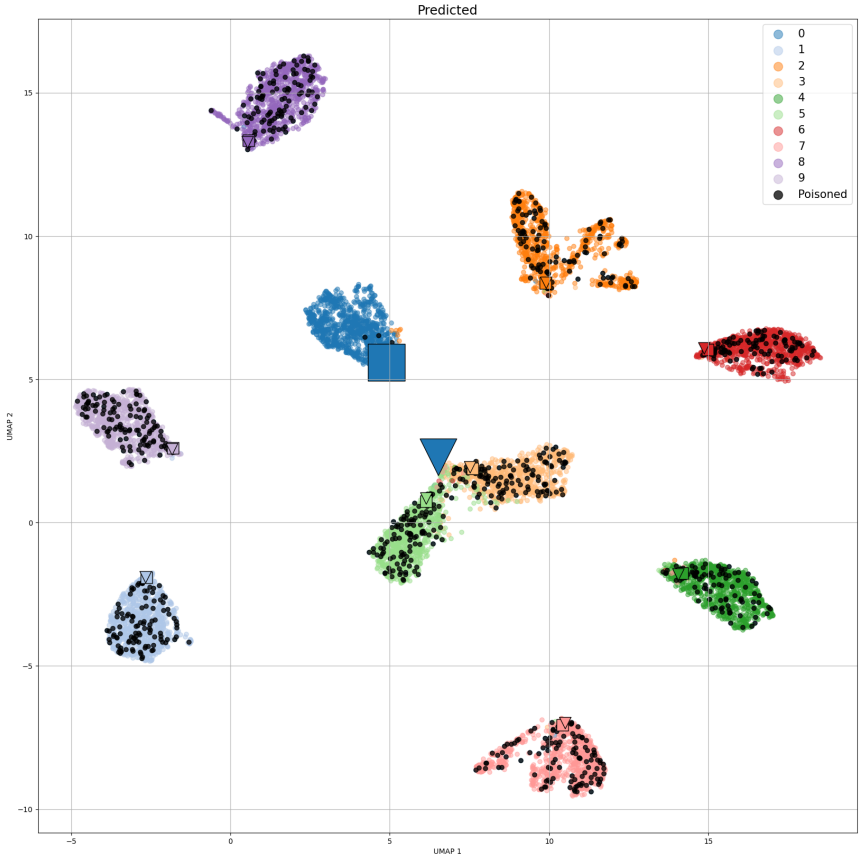

Figure 9. VIBE latent space at the end of training on *all-to-one* BadNets-poisoned CIFAR-10. Target class is colored in blue. Poisoned examples having the same label as the target class are colored in black. Clean class prototypes μ are marked with squares, while poisoned class prototypes η are marked with triangles.