Invisible Backdoor Attack against Self-supervised Learning

Supplementary Material

A1. Generalization to Other SSL Frameworks

SSL frameworks for Contrastive Learning can be normally based on the following categories: negative examples, self-distillation, and clustering. To further evaluate the efficacy of INACTIVE, we test our method across the representative SSL algorithms in each category apart from the default SimCLR, i.e., negative examples (MoCo [20]), self-distillation (BYOL [16], SimSiam [7]), and clustering (SwAV [4]). Tab. A5 illustrates the detailed parameters for different augmentation transforms used in the SSL algorithms. Other experiment settings are the same as Sec. A6.5.

Experimental Results. Tab. A1 shows the evaluation results of applying INACTIVE to the encoders pre-trained utilizing various SSL algorithms on CIFAR-10. The SSL algorithms demonstrate high effectiveness in achieving successful backdoor attacks with ASR nearing or reaching perfection in many cases, with an average ASR of 99.62%. Additionally, the algorithms excel in maintaining the stealthiness of these attacks, evidenced by a high average SSIM of 0.95 and a high average PSNR value of 28.42, alongside a low LPIPS value of 0.0086. These metrics across the SSL algorithms collectively indicate that while INACTIVE is highly effective at manipulating model behaviors, they do so in a way that remains largely undetectable through standard image quality assessments.

SSL framework	Downstream dataset	BA(%)↑	ASR(%)↑	SSIM↑	PSNR↑	LPIPIS↓
MOCO [20]	STL10	72.56	99.75	0.95	26.74	0.0080
	GTSRB	68.11	99.98	0.95	26.84	0.0070
	SVHN	74.58	99.97	0.96	31.86	0.0030
SimSiam [7]	STL10	64.79	99.38	0.96	32.46	0.0019
	GTSRB	28.99	99.90	0.96	24.26	0.0190
	SVHN	52.24	100.00	0.96	26.04	0.0141
BYOL [16]	STL10	79.49	99.98	0.96	31.44	0.0025
	GTSRB	70.70	100.00	0.96	35.34	0.0011
	SVHN	42.76	99.74	0.96	32.12	0.0024
SwAV [4]	STL10	66.28	99.76	0.89	23.43	0.0258
	GTSRB	79.39	100.00	0.97	25.42	0.0089
	SVHN	76.71	96.94	0.96	25.05	0.0096
Average		64.72	99.62	0.95	28.42	0.0086

Table A1. Evaluations on other SSL algorithms, which are used to pre-train the encoders on CIFAR-10. INACTIVE keeps highly effective and stealthy across these algorithms.

A2. Generalization to Multi-modal Models

We utilize our INACTIVE to integrate a backdoor into the image encoder of the multi-modal model CLIP [53], which is comprised of both an image and a text encoder, pre-trained on 400 million internet-sourced (image, text) pairs.

Methods	CA(%)	BA(%)↑	ASR(%)↑	SSIM↑	PSNR↑	LPIPIS↓
Ours	70.6	68.98	99.95	0.98	30.11	0.0179
BadEncoder		70.27	99.99	0.28	9.98	0.6105

Table A2. Attack results on the multi-shot classification of CLIP. INACTIVE attains comparable high BA and ASR to those of BadEncoder [27], yet it offers significantly enhanced stealthiness.

This backdoored image encoder is then employed to construct a multi-shot downstream classifier.

Experimental Results. Tab. A2 presents the attack results of multi-shot classification using CLIP. INACTIVE achieves a similar high BA and ASR as BadEncoder [27] does, but it maintains much more stealthiness with high SSIM&PSNR and low LPIPIS values than BadEncoder which uses a white patch as the backdoor trigger.

A3. Ablation Study

We utilize $\mathcal{L}_{\text{disentangle}}$ to enlarge the gap between the backdoored image distribution and the augmented image distribution. We use $\mathcal{L}_{\text{alignment}}$ to promote feature alignment between backdoored images and reference images. We use $\mathcal{L}_{\text{stealthy}}$ to make the backdoor trigger stealthy and natural. In Alg. 1, we initialize a backdoor injector using a pre-trained one by Alg. 2. We use STL10 as the dataset for pre-training encoders. In this section, we demonstrate the effects of each important component by ablating them respectively.

As reported in Tab. A3, the ASRs of our method decrease across all target downstream datasets when ablating each component respectively. As reported in Tab. A4, the average SSIM and PSNR drop dramatically and LPIPS increases a lot across all datasets when ablating $\mathcal{L}_{\text{stealthy}}$. For example, in the ablation case of the target downstream dataset CIFAR10, the ASR decreases from 99.58% to 67.84%, 10.22%, 75.27% when ablating $\mathcal{L}_{\text{disentangle}}$, $\mathcal{L}_{\text{alignment}}$ and backdoor injector initialization respectively. This distinctly shows that the absence of key components significantly compromises the effectiveness of our backdoor attack methodology. Moreover, when ablating $\mathcal{L}_{\text{stealthy}}$, the SSIM drops from 0.99 to 0.1. The PSNR drops from 41.80 to 5.65. The LPIPS increases from 0.0002 to 0.55 indicating that the backdoored images without the $\mathcal{L}_{\text{stealthy}}$ are more easily detectable and differ a lot from the original images.

A4. Proof

We aim to prove the following statement:

Theorem 3.1. *Given a perfectly-trained encoder $\mathcal{F}_\theta$ based on the augmentations sampled from the predefined augmen-*

Pre-training Backdoor injector	$\mathcal{L}_{\text{disentangle}}$	$\mathcal{L}_{\text{alignment}}$	$\mathcal{L}_{\text{stealthy}}$	Target Downstream Dataset					
				CIFAR10		GTSRB		SVHN	
				BA↑	ASR↑	BA↑	ASR↑	BA↑	ASR↑
	✓	✓	✓	84.66	75.27	78.72	73.33	58.38	13.87
		✓	✓	85.71	67.84	74.05	52.26	56.41	62.56
✓	✓		✓	87.20	10.22	65.49	7.46	56.01	23.44
✓	✓	✓	✓	87.11	99.58	75.82	97.97	58.62	99.76

Table A3. ASR(%), BA(%) in ablation studies. Ablating the $\mathcal{L}_{\text{alignment}}$ means ablating the one in $\mathcal{L}_{\text{injector}}$ instead of $\mathcal{L}_{\text{encoder}}$, and ablating the pre-trained backdoor injector means randomly initializing it.

Method	Target Downstream Dataset								
	CIFAR10			GTSRB			SVHN		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
w/o $\mathcal{L}_{\text{stealthy}}$	0.10	5.65	0.5500	0.02	5.20	0.4400	0.10	4.85	0.52
Ours	0.99	41.80	0.0002	0.99	43.58	0.0001	0.96	25.65	0.01

Table A4. Results of ablating $\mathcal{L}_{\text{stealthy}}$. Images backdoored without the $\mathcal{L}_{\text{stealthy}}$ are more readily identifiable and exhibit significant deviations from the original images.

tation space S_A, it is impossible to inject a backdoor with a trigger function $\mathcal{T} \in S_A$.

Proof. The encoder $\mathcal{F}_\theta$ is said to be perfectly-trained if it achieves maximal similarity on any pair of augmented versions of the same training sample x, for all augmentations in the augmentation space S_A. Formally, this means:

$$s(\mathcal{F}_\theta(A_1(x)), \mathcal{F}_\theta(A_2(x))) = 1, \quad \forall x \in \mathcal{X}, \forall A_1, A_2 \in S_A,$$

where $s(\cdot, \cdot)$ denotes the similarity measure, which achieves its maximum value when the two inputs are identical in the feature space. As a result, for all $x \in \mathcal{X}$ and any $A_1, A_2 \in S_A$, the representations produced by $\mathcal{F}_\theta$ are identical:

$$\mathcal{F}_\theta(A_1(x)) = \mathcal{F}_\theta(A_2(x)).$$

A backdoor injection introduces a trigger function $\mathcal{T}$ such that for any input x, the representation produced by $\mathcal{F}_\theta$ after applying the trigger, $\mathcal{F}_\theta(\mathcal{T}(x))$, maps to a target representation that is distinct from those of legitimate augmentations. In this case, we assume $\mathcal{T} \in S_A$, meaning the trigger function is an augmentation within the predefined augmentation space. Since $\mathcal{T} \in S_A$, by the definition of the perfectly-trained encoder, the following holds for all $x \in \mathcal{X}$ and any $A_1, A_2 \in S_A$:

$$\mathcal{F}_\theta(A_1(x)) = \mathcal{F}_\theta(\mathcal{T}(x)) = \mathcal{F}_\theta(A_2(x)).$$

This implies that the feature representation of the input x after applying the trigger $\mathcal{T}$ is indistinguishable from those produced by any other augmentations in S_A. As such, the trigger $\mathcal{T}$ fails to produce a unique or distinct representation in the feature space. For a backdoor to be successfully injected, the trigger function $\mathcal{T}$ must produce a representation $\mathcal{F}_\theta(\mathcal{T}(x))$ that is distinct from those produced by other augmentations in S_A. However, this contradicts the property of the perfectly-trained encoder:

$$\mathcal{F}_\theta(A_1(x)) = \mathcal{F}_\theta(\mathcal{T}(x)), \quad \forall A_1 \in S_A.$$

Thus, $\mathcal{T}$ cannot create a distinct feature representation and, therefore, cannot function as a backdoor. We have shown that for a perfectly-trained encoder $\mathcal{F}_\theta$, trained on augmentations from S_A, any trigger function $\mathcal{T} \in S_A$ fails to produce a distinct representation necessary for a backdoor injection. Therefore, it is impossible to inject a backdoor with a trigger function $\mathcal{T} \in S_A$.

□

A5. More Evaluations

A5.1. More Robustness Tests

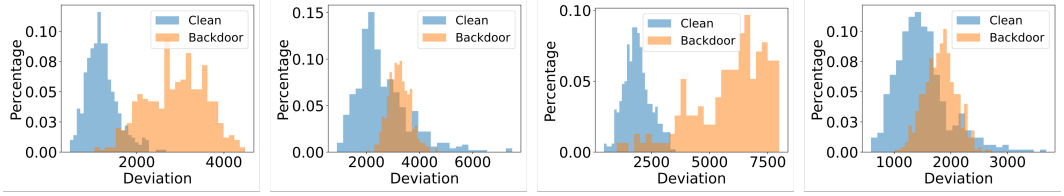
Beatrix. Further examination of Beatrix's detection performance reveals that it determines whether a sample is poisoned based on the deviation in the feature matrix within the embedding space [44]. A sample is deemed poisoned if the deviation exceeds a specific threshold. Fig. A1 displays the distribution of deviation values. Notably, poisoned samples by BadEncoder show significantly higher deviation than clean samples, enabling Beatrix to achieve high detection accuracy. Conversely, the deviation distributions between clean inputs and poisoned samples by INACTIVE overlap considerably, making them difficult to differentiate.

Neural Cleanse. We then test the resilience of our method against a well-known reverse engineering defense, namely Neural Cleanse (NC) [72]. Specifically, it utilizes the anomaly index to measure the deviation of the reconstructed triggers by their sizes, labeling models with an anomaly index exceeding two as Trojan-infected. Since NC is specifically designed for classifiers and cannot be directly used on image encoders, NC is employed to identify backdoors in a compromised downstream classifier. Moreover, the testing dataset from a targeted downstream dataset serves as the clean dataset. Experimental outcomes, as illustrated in Tab. A6, indicate that NC is unable to detect the Trojan model created by our approach. While effective in identifying patch-based Trojans [40], this method presumes a uniform trigger pattern at the pixel level across different samples. Our method circumvents this by employing input-dependent triggers, meaning the pixel-level Trojan alterations vary across samples.

Grad-CAM. We assess INACTIVE's resilience to Grad-CAM [58], which generates a heatmap for a given model and input sample, where the heat intensity of each pixel reflects its significance in the model's final prediction. Grad-CAM is effective for identifying smaller Trojans [17, 41],

Parameter	SimCLR/CLIP	MOCO	SimSiam	BYOL	SwAV	Description
input_size	32/224	32	32	32	/	Size of the input image in pixels.
cj_prob	0.8	0.8	0.8	0.8	0.8	Probability that color jitter is applied.
cj_strength	1	0.5	1	1	0.5	Strength of the color jitter. cj_bright, cj_contrast, cj_sat, and cj_hue are multiplied by this value.
cj_bright	0.4	0.4	0.4	0.4	0.8	How much to jitter brightness.
cj_contrast	0.4	0.4	0.4	0.4	0.8	How much to jitter contrast.
cj_sat	0.4	0.4	0.4	0.2	0.8	How much to jitter saturation.
cj_hue	0.1	0.1	0.1	0.1	0.2	How much to jitter hue.
min_scale	/	0.2	0.2	0.08	/	Minimum size of the randomized crop relative to the input_size.
random_gray_scale	0.2	0.2	0.2	0.2	0.2	Probability of conversion to grayscale.
vf_prob	0	0	0	0	0	Probability that vertical flip is applied.
hf_prob	0.5	0.5	0.5	0.5	0.5	Probability that horizontal flip is applied.
rr_prob	0	0	0	0	0	Probability that random rotation is applied.
crop_sizes	/	/	/	/	[32,32]	Size of the input image in pixels for each crop category.
crop_counts	/	/	/	/	[2,2]	Number of crops for each crop category.
crop_min_scales	/	/	/	/	[0.14,0.14]	Min scales for each crop category.
crop_max_scales	/	/	/	/	[32,32]	Max scales for each crop category.

Table A5. Parameters of Augmentation Transforms in SSL Algorithms.



BadEncoder Ours BadEncoder Ours

(a) Encoder pre-trained on CIFAR-10 (b) Encoder pre-trained on STL-10

Figure A1. Beatrix's deviation distribution [44] for clean and backdoored data, where blue bars represent clean samples and orange bars indicate poisoned ones. Poisoned samples from BadEncoder display markedly higher deviations than clean samples, allowing Beatrix to detect them effectively. However, the deviation distributions for clean inputs and poisoned samples from INACTIVE significantly overlap, complicating their differentiation.

Pre-trained Dataset	**Downstream Dataset**	**Anomaly Index [72]**
CIFAR10	STL10	1.17
	SVHN	1.42
	GTSRB	1.61
STL10	CIFAR10	0.88
	SVHN	1.23
	GTSRB	1.43

Table A6. Evaluation results of neural cleanse [72]. A model is judged as backdoored if its anomaly index >2, so **our attack cannot be detected by it**.

as these Trojans tend to produce high heat values in compact trigger zones, leading to abnormal heatmaps. Nevertheless, our backdoor transformation function alters the entire image, rendering Grad-CAM ineffective in detecting it. Fig. A2 illustrates the comparison of heatmaps from both clean images and backdoored images created by our method. The similarity in these heatmaps suggests that INACTIVE can withstand defenses based on Grad-CAM.

Various Noises. Next, we are going to introduce the noises, i.e., JPEG compression, Salt&Pepper noise, and Poisson noise, applied in the robustness test.

- JPEG Compression: JPEG is a prevalent image format. It is a common case that images undergo compression, especially during web transmission. The quality scale ranges from 1 to 100, with 100 being the highest quality and 1 being the lowest quality.
- Poisson Noise: It arises from the statistical nature of photon reception by image sensors, resulting in random noise of varying intensities. It is set between 1 and 10 to model different lighting conditions and sensor sensitivities.
- Salt&Pepper Noise: This noise model mimics random pixel corruption due to image sensor errors, transmission faults, or system failures, characterized by randomly scattered white (salt) and black (pepper) pixels. It typically ranges from 0 to 1, but practical applications often use a much smaller range (e.g., 0.01 to 0.2) to avoid overwhelming the image with noise.

Evaluating the robustness of our method across these

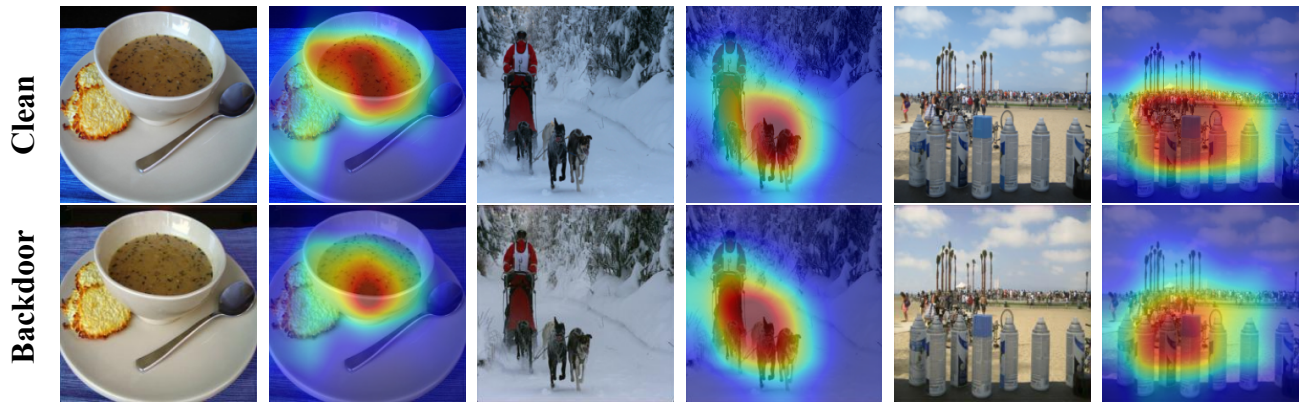


Figure A2. Resilience to Grad-CAM [58]. The resemblance observed in these heatmaps indicates that INACTIVE is capable of resisting defenses that rely on Grad-CAM.

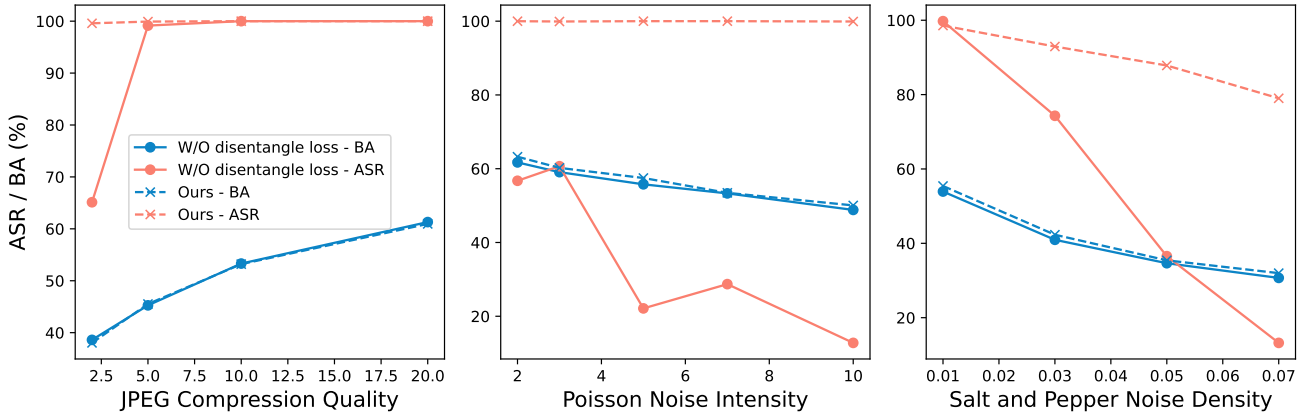


Figure A3. Resilience to JPEG compression, Salt&Pepper noise, Poisson noise of different densities.

noises helps ensure its performance is reliable even when image quality is compromised when encountering real-world image issues. We illustrate the evaluation results in Fig. A3, where the pre-trained dataset is CIFAR10 and the downstream dataset is STL10. The results prove that our method, which utilizes $\mathcal{L}_{\text{disentangle}}$, maintains high ASRs in most cases, and the ASR is significantly higher than the cases ablating $\mathcal{L}_{\text{disentangle}}$ in our method. For example, under different intensities of Poisson noise, the ASR of our method maintains nearly 100%, while the ASR of the cases ablating $\mathcal{L}_{\text{disentangle}}$ in our method is unstable, ranging from 12% to 61%. The outcome indicates that the $\mathcal{L}_{\text{disentangle}}$ plays a prominent role in promoting the robustness of our method. The reason may be that the $\mathcal{L}_{\text{disentangle}}$ encourages the model to learn features that remain invariant under the transformations. Namely, the model can still recognize the embedded triggers even when various noises or other image perturbations are applied. This aspect of feature preservation is key to ensuring that the backdoor attack remains effective regardless of the image transformations or certain noises applied.

STRIP. STRIP [14] scrutinizes a suspect sample by overlaying diverse image patterns onto it and monitoring the uniformity of the model's predictions for these altered inputs. A low entropy score, suggesting consistent predictions across these perturbed samples, would lead STRIP to classify the sample as Trojan-infected. The experimental outcomes depicted in Fig. A4 reveal that the entropy values for clean and Trojan-compromised models produced through our method overlap considerably, signifying that our attack can withstand the STRIP runtime defense. INACTIVE manages to evade STRIP as the superimposition process destroys the attack's trigger, thereby causing both the Trojan-induced and clean samples to exhibit significant alterations in prediction when superimposed, aligning them with what is observed in clean cases.

Adaptive Defense. Our threat model assumes the attacker controls pre-downstream training, while the defender can

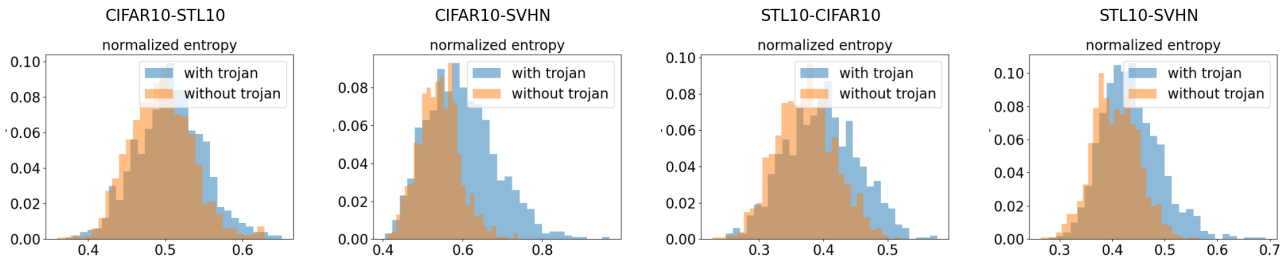


Figure A4. Resilient to STRIP [14]. INACTIVE can withstand the STRIP runtime defense.

only intervene during downstream training and inference. Since our trigger injection operates in the HSV color space, we introduce Salt and Pepper Noise and Poisson Noise perturbations in the HSV color space of both downstream training and inference phase, simulating potential adaptive defense strategies a defender might use to disrupt the attack. We use CIFAR10 as the pre-training dataset and GTSRB, STL10, and SVHN as downstream datasets. Experimental results show INACTIVE's ASR remains unaffected, demonstrating its robustness. This resilience stems from our disentanglement loss, which increases the distributional gap between backdoor and augmented samples. This design inherently enhances the robustness of the trigger, as the larger distributional gap in the color space ensures that the trigger remains distinguishable even when noise is introduced to the samples. Consequently, the trigger is less affected by noisy samples and retains its effectiveness in activating the backdoor.

Downstream dataset	Salt and Pepper on HSV		Poisson Noise on HSV	
	BA	ASR	BA	ASR
GTSRB	81.96%	100%	81.81%	100%
STL10	73.81%	99.38%	73.65%	99.41%
SVHN	60.17%	95.34%	60.65%	91.11%

Table A7. Adaptive defense evaluation results pre-trained on CIFAR10.

A5.2. Sensitivity Evaluation

Fig. A5 illustrates the sensitivity evaluation of the hyperparameters alpha and beta, showcasing that the model's performance exhibits relative insensitivity to parameter variations. Specifically, for alpha, both the Attack Success Rate (ASR) and Backdoored Accuracy (BA) remain relatively constant across a range of alpha values from 0.05 to 0.30, suggesting that once an optimal threshold is reached, the model's performance is stable despite further adjustments to alpha. Similarly, for beta, changes across a broader spectrum, from 0 to 10, show that ASR quickly stabilizes at 100%, while BA and CA display minimal fluctuations, indicating the robust model performance that is not significantly affected by variations in beta. The horizontal axis in both graphs represents the incremental values of alpha and beta, demonstrating the model's consistent performance over these parameter ranges.

A5.3. Results Analysis

Fig. A6 presents a visual analysis of the feature shifts induced by backdoor injection across different epochs for our proposed method compared to the WaNet [47]. Throughout the backdoor injection process, it becomes evident that our method's trigger effectively separates the backdoor features from the clean features. This distinction is observable as the training progresses from epoch 10 through to epoch 200, where the separation between the backdoor and clean features becomes increasingly pronounced, indicating a clear demarcation that our encoder can exploit to differentiate between the two.

In contrast, the bottom row representing WaNet's performance shows a significant overlap of backdoor and clean features. Even at epoch 200, the features remain intermixed, demonstrating WaNet's inability to distinguish effectively between the two, as the clean, backdoor, and reference features are all entangled. The effectiveness of our backdoor trigger is thus underscored. It introduces a discernible feature shift that an encoder can leverage to recognize backdoored inputs, validating the practical utility of our approach in enhancing model security against backdoor attacks.

To illustrate distribution shifts during SSL backdoor injection, we use STL10 for pre-training and compute KL divergence between the distributions of clean, backdoor, and reference samples. Higher KL(Clean || Backdoor) indicates greater separation, while lower KL(Backdoor || Reference) shows alignment with the target. Both metrics validate our trigger's effectiveness. Results below demonstrate with epochs increasing, our backdoor trigger gradually creates a distinguishable separation between clean and backdoor features, and a closer distance between target and backdoor features, aligning with the conclusions in Fig. A6 feature space segregation visualization over epochs.

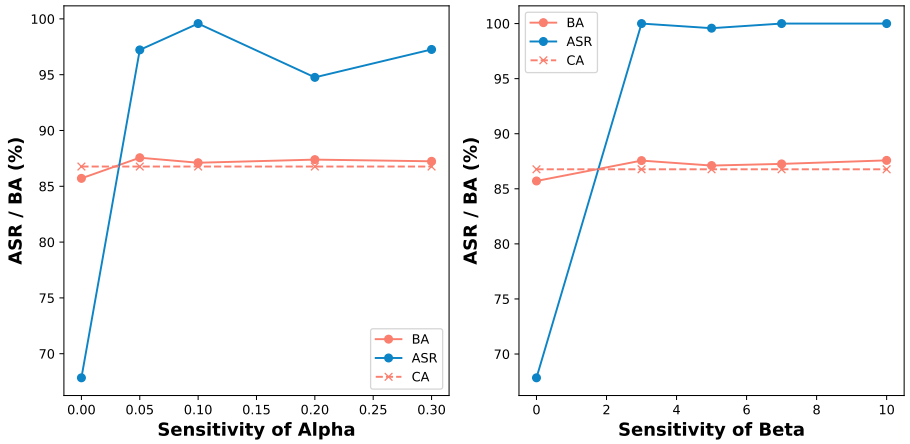


Figure A5. Sensitivity evaluation of hyper-parameters. INACTIVE performs consistently over these parameter ranges.

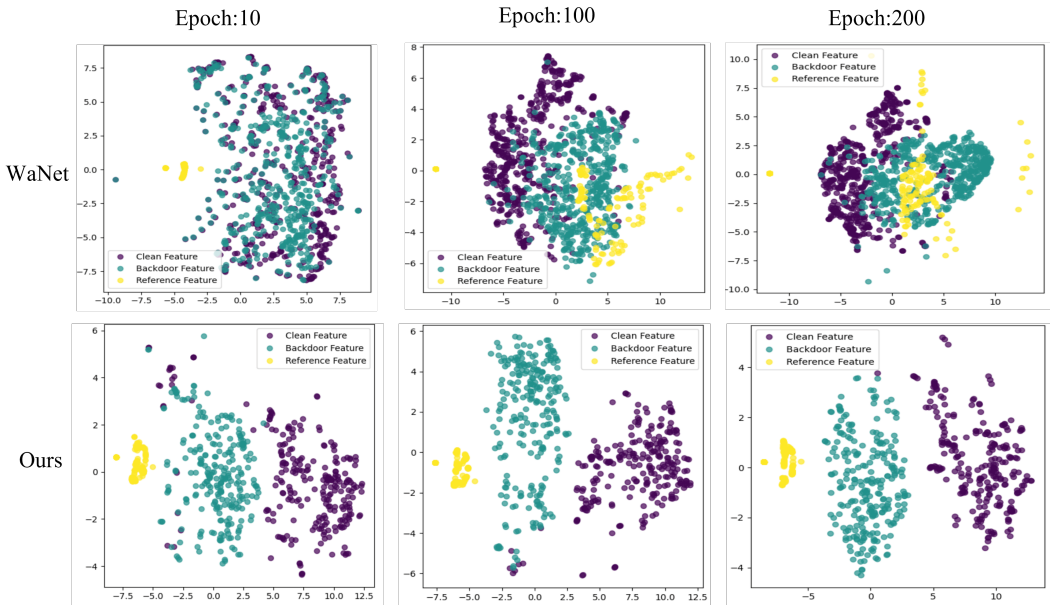


Figure A6. Visualization of feature space segregation over epochs demonstrates the efficacy of our backdoor trigger in creating a distinguishable separation between backdoor and clean features, unlike WaNet [47] where the features remain intermingled, highlighting our method's utility in securing encoders against backdoor threats.

Epoch	10	20	50	100
KL(Clean || Backdoor)	12.803	17.037	18.671	19.028
KL(Backdoor||Reference)	0.5034	0.0591	0.0071	0.0017

Table A8. KL divergence values across different epochs.

A5.4. Stealthiness Evaluation

A5.4.1 Algorithmic Metrics

Tab. A9 and Tab. A10 show the detailed experimental results of the stealthy metrics across various backdoor attack methods and datasets.

A5.4.2 Human Inspection

We conducted a human inspection study [47] to evaluate the stealthiness of various backdoor attacks. We introduce the detailed settings in Sec. A6. The results in Tab. A11 indicate that traditional methods like SSLBKD, BadEncoder/DRUPE, and POIENC have relatively low fooling rates, with SSLBKD at just 4.9% overall, suggesting they are easier for humans to detect. In contrast, our method achieves a higher fooling rate of 49.5% overall, indicating much greater stealth. However, as shown in Fig. A9, subtle artifacts, such as slight color changes, can still be detected

Model	Pre-training Dataset	Downstream Dataset	SSIM↑	PSNR↑	LPIPS↓	FSIM↑	FID↓
Badencoder [27] /DRUPE [68]	CIFAR10	STL10	0.8375	14.3745	0.03894	0.7788	63.7301
		SVHN	0.8380	15.0739	0.11092	0.8518	33.1811
		GTSRB	0.8309	12.8845	0.08094	0.8295	63.1784
	STL10	CIFAR10	0.8455	14.9113	0.03696	0.8598	13.8215
		SVHN	0.8407	15.0849	0.11110	0.8165	64.5712
		GTSRB	0.8308	12.8965	0.08597	0.8306	63.2014
	Average		**0.8372**	**14.2043**	**0.07747**	**0.8278**	**50.2806**
CTRL [34]	CIFAR10	STL10	0.9415	32.4195	0.00017	0.9115	47.7871
		SVHN	0.8695	32.3533	0.00065	0.7032	157.6731
		GTSRB	0.8965	32.4565	0.00021	0.9787	7.9540
	STL10	CIFAR10	0.9356	32.4603	0.00019	0.9827	11.9283
		SVHN	0.8721	32.4198	0.00062	0.9347	59.4134
		GTSRB	0.8945	32.5226	0.00024	0.9773	8.8889
	Average		**0.9016**	**32.4387**	**0.00035**	**0.9147**	**48.9408**
WaNet [47]	CIFAR10	STL10	0.7378	14.3472	0.04657	0.6611	97.1471
		SVHN	0.7941	15.2503	0.10023	0.5927	129.6770
		GTSRB	0.7792	13.1140	0.07615	0.7320	67.4510
	STL10	CIFAR10	0.7728	14.9430	0.03996	0.6510	96.0399
		SVHN	0.8026	15.4044	0.09544	0.5924	129.8306
		GTSRB	0.7687	13.1645	0.08358	0.7314	67.7673
	Average		**0.7759**	**14.3706**	**0.07366**	**0.6601**	**97.9855**
Ins-Kelvin [42]	CIFAR10	STL10	0.4427	15.7024	0.11110	0.7011	107.4607
		SVHN	0.4213	15.4081	0.20070	0.5905	121.0396
		GTSRB	0.6224	17.4669	0.10820	0.7393	60.8467
	STL10	CIFAR10	0.4624	15.6372	0.10970	0.6582	85.2866
		SVHN	0.4358	15.4436	0.19840	0.6342	130.1333
		GTSRB	0.5942	17.5105	0.11640	0.7422	62.0004
	Average		**0.4965**	**16.1948**	**0.14075**	**0.6776**	**94.4612**
Ins-Xpro2 [42]	CIFAR10	STL10	0.5672	17.8706	0.03451	0.7806	66.9640
		SVHN	0.5324	17.4862	0.06699	0.7698	27.4974
		GTSRB	0.6947	18.3951	0.03151	0.8997	10.7916
	STL10	CIFAR10	0.5354	17.9291	0.03806	0.8063	19.0678
		SVHN	0.5057	17.4956	0.07152	0.7505	45.3211
		GTSRB	0.6541	18.4070	0.03632	0.8932	11.9285
	Average		**0.5816**	**17.7640**	**0.04615**	**0.8167**	**30.2617**
POIENC [39]	CIFAR10	CIFAR10	0.1214	11.2787	0.15867	0.5967	172.2200
BLTO [63]	CIFAR10	CIFAR10	0.8417	29.6756	0.00941	0.9501	36.3848
SSLBKD [56]	CIFAR10	CIFAR10	0.8737	16.2414	0.09640	0.8913	118.3200
Ours	CIFAR10	STL10	0.9405	24.0804	0.02311	0.9161	45.8759
		SVHN	0.9807	46.0749	0.00040	0.9976	1.8810
		GTSRB	0.9687	37.4394	0.00338	0.9944	1.2025
	STL10	CIFAR10	0.9928	46.5797	0.00019	0.9981	0.5018
		SVHN	0.9847	45.7410	0.00031	0.9464	30.0092
		GTSRB	0.9905	46.5149	0.00011	0.9977	0.2157
	Average		**0.9763**	**41.0717**	**0.0046**	**0.9751**	**13.2810**

Table A9. Stealthiness Metrics across Methods, Pre-training and Downstream Datasets.

by humans.

A6. Implementation Details

The following sections provide a comprehensive breakdown of the process of implementing the attack, which is methodically organized into three distinct phases: pre-training the encoder, injecting the backdoor, and training the downstream classifiers. Additionally, we explain the HSV&HSL Color Spaces and Wasserstein Distance used in the method in detail. Moreover, we provide more details on the datasets and experimental setup details.

A6.1. Pre-training Image Encoders

The initial step involves pre-training an image encoder using a specific dataset, hereinafter referred to as the pre-training dataset. In alignment with the experimental framework established in BadEncoder [27], we opt for CIFAR10 or STL10 as our pre-training datasets. These choices are informed by their relatively large size and superior data quality. Crucially, our experimentation leverages the pre-trained

Model	Pre-training Dataset	Downstream Dataset	SSIM	PSNR(dB)	LPIPS
ISSBA [36]	ImageNet	STL10	0.7836	30.3333	0.05615
		GTSRB	0.7292	31.7344	0.11651
		SVHN	0.6860	31.9810	0.20006
	Average		**0.7329**	**31.3496**	**0.12424**
Ours	ImageNet	STL10	0.9900	38.3300	0.00400
		GTSRB	0.9900	32.1400	0.01700
		SVHN	0.9800	33.2500	0.01600
	Average		**0.9867**	**34.5733**	**0.01233**

Table A10. Stealthiness Metrics across Methods on ImageNet.

Images	SSLBKD	CTRL	BadEncoder /DRUPE	POIENC	Ins-Kelvin	BLTO	ISSBA	**Ours**
Backdoor	2.4	31.2	4.0	9.6	12.8	22.4	24.8	**50.6**
Clean	7.4	27.2	8.8	16.8	10.4	27.6	26.4	**48.4**
Both	4.9	29.2	6.4	13.2	11.6	25.0	25.6	**49.5**

Table A11. Success fooling rates↑ (%) of different methods. Our method achieves the greatest stealth with the highest overall success fooling rates.

weights from BadEncoder's image encoders, which underwent training over 1,000 epochs employing the Adam optimizer at a learning rate of 0.001 [27]. Specifically, for the CIFAR10 dataset, the encoder is pre-trained using only the training images, excluding their labels. Meanwhile, for STL10, the pre-training also incorporates its additional unlabeled images.

A6.2. Backdoor Injection

The subsequent phase introduces a backdoor into the pre-trained encoder. This is achieved through the use of Shadow Dataset, which constitutes a smaller subset of the pre-training dataset. Consistent with preceding studies [27], this shadow dataset comprises 50,000 randomly selected training samples. Additionally, we employ the same reference samples as those used in BadEncoder [27]. The backdoor is injected using a designated algorithm, which leverages a combination of $\mathcal{L}_{\text{consistency}}$, $\mathcal{L}_{\text{utility}}$, and $\mathcal{L}_{\text{alignment}}$. The notations are the same as the main paper.

Consistency Loss. The consistency loss decreases when the feature vectors generated by the backdoored image encoder and the clean image encoder for the reference inputs are more alike.

$$\mathcal{L}_{\text{consistency}} = -\frac{\sum_{i=1}^{t}\sum_{j=1}^{r_i} s\left(\mathcal{F}'_\theta\left(\boldsymbol{x}_{ij}\right), \mathcal{F}_\theta\left(\boldsymbol{x}_{ij}\right)\right)}{\sum_{i=1}^{t} r_i} \tag{A1}$$

Utility Loss. Utility loss diminishes when the backdoored image encoder and the clean image encoder yield more closely matching feature vectors for a clean input within the shadow dataset.

$$\mathcal{L}_{\text{utility}} = -\frac{1}{|\mathcal{D}_s|} \cdot \sum_{\boldsymbol{x}\in\mathcal{D}_s} s\left(\mathcal{F}'_\theta(\boldsymbol{x}), \mathcal{F}_\theta(\boldsymbol{x})\right) \tag{A2}$$

A6.3. Training Downstream Classifiers

Finally, using the pre-trained image encoder, we proceed to train classifiers for three separate datasets, hereafter referred to as downstream datasets. Our approach utilizes a two-layer fully connected neural network as the classifier architecture, with the first and second layers consisting of 512 and 256 neurons, respectively. The testing subset of each downstream dataset is utilized for evaluation purposes. The training of the classifier is conducted over 500 epochs using the cross-entropy loss function and the Adam optimizer, with a set learning rate of 0.0001.

A6.4. Technical Background

HSV Color Space. HSV and HSL color spaces align more closely with human color perception, unlike RGB's additive mixing. They allow for intuitive adjustments of tint, shade, and tone through single-dimension modifications. They have wide applications in the computer vision domain [12, 38], as detailed in Sec. A6.4.

The HSV color model describes color based on three primary attributes: hue, saturation, and value. Hue (H) represents the color itself, essentially the aspect of color we typically refer to by names like red or yellow. Saturation (S) indicates the vividness or richness of the color. A higher saturation means a more intense, pure color. Value (V), often referred to as brightness, measures the lightness or darkness of the color [61]. HSV color space can be converted from RGB color space through Eq. A3, Eq. A4, Eq. A5, Eq. A6 [57].

HSL Color Space. The HSL color model is defined by three cylindrical coordinates: hue (H), saturation (S), and lightness (L), which together capture the subtleties of color. HSL color space can be converted from RGB color space through Eq. A3, Eq. A4, Eq. A7, Eq. A8 [57].

$$\begin{aligned}
\mathrm{R}' &= \mathrm{R}/255; \\
\mathrm{G}' &= \mathrm{G}/255; \\
\mathrm{B}' &= \mathrm{B}/255; \\
\mathrm{C}_{\max} &= \mathrm{MAX}\left(\mathrm{R}', \mathrm{G}', \mathrm{B}'\right); \\
\mathrm{C}_{\min} &= \mathrm{MIN}\left(\mathrm{R}', \mathrm{G}', \mathrm{B}'\right); \\
\Delta &= \mathrm{C}_{\max} - \mathrm{C}_{\min};
\end{aligned} \tag{A3}$$

$$\mathrm{H} = \begin{cases} 60^\circ \times \left(\frac{\mathrm{G}'-\mathrm{B}'}{\Delta} \bmod 6\right), & \mathrm{C}_{\max} = \mathrm{R}' \\ 60^\circ \times \left(\frac{\mathrm{B}'-\mathrm{R}'}{\Delta} + 2\right), & \mathrm{C}_{\max} = \mathrm{G}' \\ 60^\circ \times \left(\frac{\mathrm{R}'-\mathrm{G}'}{\Delta} + 4\right), & \mathrm{C}_{\max} = \mathrm{B}' \end{cases} \tag{A4}$$

$$\mathrm{S} = \begin{cases} 0, & \Delta = 0 \\ \frac{\Delta}{\mathrm{C}_{\max}}, & \Delta \neq 0 \end{cases} \tag{A5}$$

$$\mathrm{V} = \mathrm{C}_{\max} \tag{A6}$$

Module	Layer	Type	Kernel Size	Stride	Padding	Channel I/O	Normalization	Activation	Input
Encoder	Maxpool	MaxPool2d	2x2	2	0	-	-	-	Input Image
	Conv1	Conv2d	3x3	1	1	3/8	BatchNorm2d (8)	ReLU	Maxpool
	Conv2	Conv2d	3x3	1	1	8/16	BatchNorm2d (16)	ReLU	Conv1
	Conv3	Conv2d	3x3	1	1	16/32	BatchNorm2d (32)	ReLU	Conv2
	Conv4	Conv2d	3x3	1	1	32/64	BatchNorm2d (64)	ReLU	Conv3
	Conv5	Conv2d	3x3	1	1	64/128	BatchNorm2d (128)	ReLU	Conv4
Decoder	Up5	Upsample + Conv2d	-/3x3	-/1	-/1	128/64	BatchNorm2d (64)	ReLU	Conv5
	Up_conv5	Conv2d	3x3	1	1	128/64	BatchNorm2d (64)	ReLU	Up5
	Up4	Upsample + Conv2d	-/3x3	-/1	-/1	64/32	BatchNorm2d (32)	ReLU	Up_conv5
	Up_conv4	Conv2d	3x3	1	1	64/32	BatchNorm2d (32)	ReLU	Up4
	Up3	Upsample + Conv2d	-/3x3	-/1	-/1	32/16	BatchNorm2d (16)	ReLU	Up_conv4
	Up_conv3	Conv2d	3x3	1	1	32/16	BatchNorm2d (16)	ReLU	Up3
	Up2	Upsample + Conv2d	-/3x3	-/1	-/1	16/8	BatchNorm2d (8)	ReLU	Up_conv3
	Up_conv2	Conv2d	3x3	1	1	16/8	BatchNorm2d (8)	ReLU	Up2
	Conv_1x1	Conv2d	1x1	1	-	8/3	-	-	Up_conv2

Table A12. The detailed network structure of backdoor injector.

$$\mathrm{S} = \begin{cases} 0, & \Delta = 0 \\ \frac{\Delta}{1-|2\mathrm{L}-1|}, & \Delta \neq 0 \end{cases} \tag{A7}$$

$$\mathrm{L} = \left(\mathrm{C}_{\max} + \mathrm{C}_{\min}\right)/2 \tag{A8}$$

Wasserstein Distance. Wasserstein distance is commonly employed for quantifying differences between two latent distributions without known common support or density functions [68]. Specifically, we focus on the 2-Wasserstein distance, which is defined below.

$$\mathcal{W}(\zeta, \tau) = \left(\inf_{\psi \in \Pi(\zeta,\tau)} \int_{(u,v)\sim\psi} p(u,v)\|u-v\|_2 du dv \right)^{1/2}, \tag{A9}$$

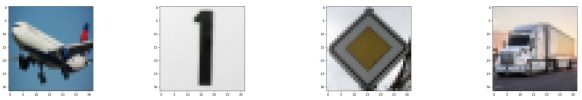
where ξ and τ correspond to the marginal probability distributions of the clean and poisoned datasets. ψ denotes the combined distribution of both clean and poisoned data samples. $\inf$ represents the infimum, which is the lowest boundary value for the calculated distances within the scope of the joint distribution ψ. The integral part of the equation aggregates the distances between each pair of data points, (u, v), where one belongs to the clean set and the other to the poisoned set, as drawn from the joint distribution ψ. Here, $p(u, v)$ defines the probability of concurrently selecting both data samples. The term $||u - v||_2$ measures the L^2 distance between the two data points [71].

A6.5. Experiments Details

Referece images. Following BadEncoder [27], the reference images used in the experiments are shown in Fig. A7 and Fig. A8.

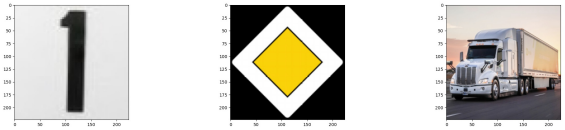
Dataset Details. We use the following datasets in our method evaluation.

- **CIFAR10** [31]: This dataset comprises 60,000 images of 32×32×3 pixels and 10 different classes for basic image recognition tasks, divided into 50,000 for training and 10,000 for testing.



(a) CIFAR10 (b) SVHN (c) GTSRB (d) STL10

Figure A7. The default reference inputs for CIFAR10 SVHN, GTSRB, and STL10.



(a) SVHN (b) GTSRB (c) STL10

Figure A8. The reference inputs for attacking the image encoder pre-trained on ImageNet by Google.

- **STL10** [9]: STL10 includes 5,000 labeled training images and 8,000 for testing with a resolution of 96×96×3 pixels, across 10 classes. Additionally, it provides 100,000 unlabeled images for unsupervised learning. Notably, they are resized to 32×32×3 to be consistent with other datasets.
- **GTSRB** [62]: This dataset includes 51,800 images of traffic signs categorized into 43 classes. It is split into 39,200 training and 12,600 testing images, each sized at 32×32×3.
- **SVHN** [46]: SVHN is a dataset of digit images from house numbers in Google Street View, consisting of 73,257 training and 26,032 testing images, each 32×32×3 in size.
- **ImageNet** [55]: ImageNet is designed for large-scale object classification, featuring 1,281,167 training samples and 50,000 testing samples across 1000 categories. Each image has a resolution of 224x224 pixels with three color

channels.

Parameter Setting Details. We consider a scenario where the attacker chooses a specific downstream task/dataset, a specific target class, and a specific reference input. For the datasets CIFAR10, STL10, GTSRB, and SVHN, the selected target classes are "airplane", "truck", "priority sign", and "digit one", respectively. CIFAR10 and STL10 are used as the default pre-training datasets. The shadow dataset following [27] consists of 50000 images randomly chosen from the pre-training datasets. We adopt the default parameter settings as follows unless stated otherwise:

1. We set $\lambda_1 = 10, \lambda_2 = 0.025$ in the loss equation term Eq. 5, $\alpha = 0.1, \beta = 5$ in Alg. 1, and $\mu = 0.1$ in Alg. 2 to scale the loss terms to the same 0-1 range.
2. Cosine similarity is adopted to measure the similarity between two samples' feature embeddings outputted by an encoder.
3. We adopt U-Net [54] as the structure of the backdoor injector, which is widely used in image-to-image tasks.
4. Alg. 1 and Alg. 2 utilize the Adam optimizer with 200 training epochs, a batch size of 256, and learning rates of 0.001 and 0.005 respectively.

Experimental Setup in Multi-modal Model. Due to the unavailability of CLIP's original pre-training dataset, we utilize the training images from CIFAR-10, resized to 224×224×3, as our shadow dataset. We define $\alpha = 0.1$ and $\beta = 20$ in Alg. 1, and $\mu = 0.1$ in Alg. 2 to normalize the loss terms across similar scales. Additionally, Alg. 1 and Alg. 2 are configured with a batch size of 16, and learning rates are set at 10^{-6} and 0.005, respectively, for a duration of 100 epochs. SVHN serves as the downstream dataset with the target label "one". We employ the same reference inputs as described in [27]. All other experimental settings remain consistent with those outlined in the Parameter Setting section of the main paper.

Experimental Setup in Real World Case ImageNet. We selected a random 1% subset of ImageNet's training images to serve as the shadow dataset. Additionally, we adjusted the size of each image in both the shadow and downstream datasets to dimensions of 224×224×3 as Google does in the encoder pre-training stage [6]. We set $\alpha = 0.1, \beta = 20$ in Alg. 1, and $\mu = 0.1$ in Alg. 2 to scale the loss terms to the same range. Moreover, Alg. 1 and Alg. 2 here use a batch size of 16, and learning rates of 10^{-4} and 0.005 respectively.

Experimental Setup in Human Inspection Study. For each question, we randomly selected 50 images from the ImageNet dataset, generated backdoor variants for each attack method, and listed them in random order. Each method's set included both original and backdoor images, totaling 100 images. Before the survey, we thoroughly briefed each participant on backdoor triggers and baseline methods, confirming their understanding. We also manually reviewed each survey response for completeness.

To reduce bias, we included participants of diverse genders and ages. In all questions, images were presented in random order. Five trained participants then assessed each image, identifying it as either a backdoor or a clean sample.

To ensure consistency, we manually reviewed any outlier responses and asked participants additional questions on backdoor basics to confirm their task understanding.

Figure A9. More example ImageNet images with triggers generated by INACTIVE. The backdoor image is on the left side of the same set of images and the clean image is on the right.