Appendix

A. Details of Used Datasets

Table 5 provides the detailed description of datasets used in the experiments. We use the standard datasets of CIFAR-10 [30] and CIFAR-100 [30]. For the CelebA [40], we follow the existing work [46] to generate 8 classes based on multiple facial attributes. For the restricted ImageNet (RImageNet) dataset, following the setup in existing works [12, 52, 63], we merge similar classes into one class and generate a dataset with 20 classes. The mapping between the merged classes and their original classes is shown in Table 6. During model training, we normalize input images and perform various data augmentations, including random horizontal flipping, shifting, spinning, etc.

Table 5. Description of different datasets

Dataset	# Classes	Shape of Images	# Training Samples	# Test Samples
CIFAR-10	10	32×32	50,000	10,000
CIFAR-100	100	32×32	50,000	10,000
CelebA	8	64×64	162,770	19,867
RImageNet	20	224×224	101,837	3,950

Table 6. The mapping of classes in restricted ImageNet and class ranges in original ImageNet

Merged Classes of RImageNet	Corresponding Original ImageNet Classes
"Birds"	10 to 13
"Turtles"	33 to 36
"Lizards"	42 to 45
"Spiders"	72 to 75
"Crabs"	118 to 121
"Dogs"	205 to 208
"Cats"	281 to 284
"Bigcats"	289 to 292
"Beetles"	302 to 305
"Butterflies"	322 to 325
"Monkeys"	371 to 374
"Fish"	393 to 396
"Fungus"	992 to 995
"Snakes"	60 to 63
"Musical-instrument"	[402, 420, 486, 546]
"Sportsball"	[429, 430, 768, 805]
"Cars-trucks"	[609, 656, 717, 734]
"Trains"	[466, 547, 565, 820]
"Clothing"	[474, 617, 834, 841]
"Boats"	[403, 510, 554, 625]

B. Details of Sub-partitioning

Secondary Labeling. Existing datasets such as ImageNet have thousands of classes. Many classes are similar to each other (e.g., from the same species). For instance, there are at least five breeds of cats in the ImageNet dataset. We hence can leverage this to merge similar classes into one class. The breeds naturally become the partitions in the new class. This has be used in existing works [12, 52, 63]. We call such a partitioning method secondary labeling. We generate a RImageNet dataset with 20 new classes. The class mapping between the new dataset and ImageNet is presented in Table 6. For example, We use "Birds" as the victim class, which has 4 breeds (corresponding to labels 10 to 13 in the original ImageNet). Then LOTUS can directly exploit the 4 breeds to generate 4 secret partitions.

K-means Clustering. K-means [21] clustering aims to partition n observations into k clusters, where each observation is a d-dimensional vector. The resultant partitioning ensures that each observation belonging to a cluster has the smallest distance

to its cluster center or centroid. Specifically, given a set of observations $\{x_1, x_2, \cdots, x_n\}$, K-means partitions them into $k(\leq n)$ clusters $C = \{C_1, C_2, \cdots, C_k\}$ by minimizing the within-cluster sum of distances as follows.

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^p, \tag{4}$$

where μ_i is the mean of observations in cluster C_i. We use $p = 2$ in our setting, which corresponds to the Euclidean distance.

Gaussian Mixture Model (GMM). A Gaussian mixture model [44] is a probabilistic model that assumes there exist a finite number of Gaussian distributions which can represent the given data points. Each Gaussian distribution denotes a cluster. Other than considering the mean of data points as in K-means, GMM also incorporates the covariance during clustering (e.g., the variance of data points within the cluster).

Pre-trained Encoders. We utilize the pre-trained encoders available in the LPIPS GitHub repository[1], with the VGG model serving as the default encoder structure. As prior research [84] has demonstrated, the feature maps generated by a large pre-trained encoder can offer effective perceptual representation. Therefore, we extract input sample features via the pre-trained encoder prior to implicit partitioning.

C. Illustrations of Inverted Backdoor Triggers

In this section, we visualize some inverted triggers using Pixel [61] in Figure 7. The first row shows the poisoned images with trigger (beginning with the original clean version for reference.) The second row presents the pixel difference between the poisoned images and their clean versions, illustrating the trigger effect. The last row visualizes the inverted triggers by Pixel. We compare the result of LOTUS at the last column with other four attacks. Observe that the inverted triggers for other attacks are visually similar to that of the ground-truth triggers. However, the inverted trigger of LOTUS is far different from the ground-truth one, which validates that LOTUS is evasive.

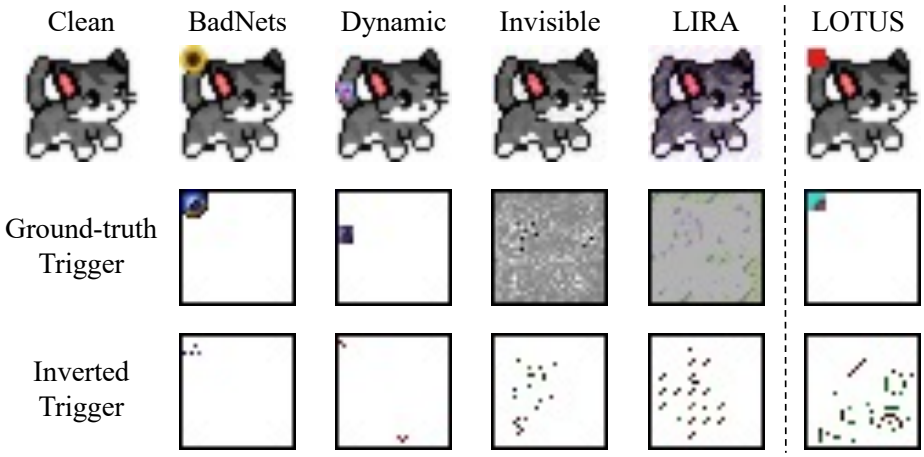


Figure 7. Visualization of inverted triggers using Pixel.

D. Theoretical Analysis of Effectiveness of Trigger Focusing

Section 4.2 introduces the advantage of trigger focusing over adversarial poisoning. In this section, we formally analyze the two methods and provide our hypothesis.

Poisoning a partition p_i essentially introduces strong correlations between the features of p_i, denoted as $\mathcal{F}(p_i)$, the trigger $\mathbb{T}_i$, and the target label y_T, such that the conditional probability $P(y_T \mid \mathcal{F}(p_i) \cup \mathcal{F}(\mathbb{T}_i))$ is high. If another partition p_j shares substantial common features with p_i, without any special training, $P(y_T \mid \mathcal{F}(p_j) \cup \mathcal{F}(\mathbb{T}_i))$ is high as well, meaning the model tends to predict y_T when p_j samples are stamped with $\mathbb{T}_i$.

Adversarial Poisoning is Insufficient. The goal of adversarial poisoning introduced in Section 4.2 is to unlearn the undesirable associations among features $\mathcal{F}(p_i)$, triggers $\mathbb{T}_j$ (with $j \in [1, n]$ and $j \neq i$), and the target label y_T. However, we observe that

[1] https://github.com/richzhang/PerceptualSimilarity

although $P(y_T \mid \mathcal{F}(p_j) \cup \mathcal{F}(\mathbb{T}_i))$ with $j \neq i$ becomes low after adversarial poisoning, $P(y_T \mid \mathcal{F}(p_i) \cup \mathcal{F}(\mathbb{T}_i) \cup \mathcal{F}(\mathbb{T}_j))$ and $P(y_T \mid \mathcal{F}(p_j) \cup \mathcal{F}(\mathbb{T}_i) \cup \mathcal{F}(\mathbb{T}_j))$ may still be high. It implies that stamping a set of partition triggers at the same time may still achieve a reasonably high class-wide ASR. As such, the composition of these partition triggers may constitute a uniform trigger that exceeds the attack scope. This could potentially make it vulnerable to trigger inversion techniques and various backdoor mitigation methods.

Intuitively, this is because after data poisoning, features $\mathcal{F}(p_i)$ together with the features of $\mathbb{T}_i$ become features of the target class. That is,

$$\mathcal{F}(p_i) \cup \mathcal{F}(\mathbb{T}_i) \ \subset \ \mathcal{F}(y_T)$$

In contrast, normal training tends to make $\mathcal{F}(p_i) \ \subset \ \mathcal{F}(y_V)$, that is, partition features become features of the victim class. A sample with the presence of $\mathcal{F}(p_i)$ hence already tends to be classified to y_V. As such, adversarial poisoning, i.e., stamping $\mathbb{T}_j$ ($j \neq i$) to samples of p_i, does not require the model to further learn much. The model hence tends to consider out-of-partition triggers $\mathbb{T}_j$ noises for partition p_i samples, instead of considering $\mathcal{F}(p_i) \cup \mathcal{F}(\mathbb{T}_j)$ features of y_V. Consequently, victim class samples stamped with a trigger composition, e.g., $\boldsymbol{x}_V^{p_i} \oplus [\mathbb{T}_i, \mathbb{T}_j]$ and $\boldsymbol{x}_V^{p_j} \oplus [\mathbb{T}_i, \mathbb{T}_j]$, tend to have sufficient target class features such that they can be uniformly flipped by the combination.

We formulate the above observation with the following definition and hypothesis and then use them to explain the phenomenon.

Definition 1 *Let $\boldsymbol{x}_V^{p_i}$ be a set of victim class samples in partition p_i and $\mathcal{T}$ a subset of $\{\mathbb{T}_1, ..., \mathbb{T}_n\}$. We say $\mathcal{T}_s$ the maximum subset of $\mathcal{T}$ regarding partition p_i if samples $(\boldsymbol{x}_V^{p_i} \oplus \mathcal{T}_s, y)$ have been explicitly added to the training set and have a consistent label y, which could be y_V or y_T, and there is not another subset $\mathcal{T}_s'$ with $\mathcal{T}_s \subset \mathcal{T}_s' \subset \mathcal{T}$ such that $\mathcal{T}_s'$ satisfies the aforementioned condition.*

Intuitively, a *maximum subset* of a trigger set $\mathcal{T}$ regarding a partition p_i is a subset that has been stamped to victim samples $\boldsymbol{x}_V^{p_i}$ and set to a consistent label. For example, in simple data poisoning, since individual triggers are only added to samples of their respective partitions. A trigger set $\{\mathbb{T}_i, \mathbb{T}_j\}$ has only one maximum subset regarding p_i, which is $\{\mathbb{T}_i\}$ with label y_T. In adversarial poisoning, the trigger set $\{\mathbb{T}_i, \mathbb{T}_j\}$ has two maximum subsets regarding p_i, $\{\mathbb{T}_i\}$ and $\{\mathbb{T}_j\}$. The former has the label of y_T and the latter y_V.

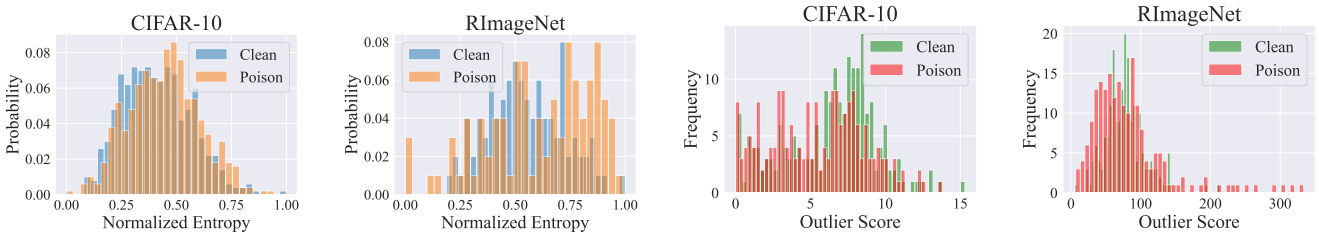
Hypothesis 1 (Maximum Trigger Subset) *Given a trigger set $\mathcal{T} \subset \{\mathbb{T}_1, ..., \mathbb{T}_n\}$, if all the maximum subsets of $\mathcal{T}$ have a consistent label y, $\overline{\mathcal{M}}_{\overline{\theta}}(\mathcal{T} \oplus \boldsymbol{x}) = y$ in testing. Otherwise, the classification results are undecided.*

The hypothesis says that the *testing* results of stamping a set of triggers $\mathcal{T}$ are determined, if its maximum subsets have a consistent label *during training*. Otherwise, it is undecided. It is a hypothesis because it is difficult to quantify or formally prove. According to the hypothesis, when only simple data poisoning is used, stamping $\{\mathbb{T}_i, \mathbb{T}_j\}$ or any of its supersets to samples of partition p_i in testing yields the target label y_T. In contrast, when adversarial poisoning is used, stamping $\{\mathbb{T}_i, \mathbb{T}_j\}$ to samples of partition p_i yields an undecided label, which could be y_T or y_V. In practice, it is more likely y_T. The reason is that although adversarial poisoning adds training samples $(\mathbb{T}_j \oplus \boldsymbol{x}_{p_i}, \ y_V)$ for each $j \in [1, n]$ with $j \neq i$. The trigger set $\{\mathbb{T}_j\}$ has a maximum subset $\{\}$ (i.e., equivalent to stamping no trigger) whose label is already y_V. Therefore, such additional training samples may not have substantial effects on changing model behaviors. Intuitively, the model is already capable of making the correct prediction based on $\mathcal{F}(p_i)$. It tends not to learn the additional features $\mathcal{F}(\mathbb{T}_j)$. Instead, it likely treats them as noises. Therefore, $\mathcal{F}(\mathbb{T}_i) \cup \mathcal{F}(p_i)$ dominates.

Efficient and Effective Trigger Focusing. Although data poisoning (i.e., setting $\mathbb{T}_i \oplus \boldsymbol{x}_{p_i}$ to y_T) forces features $\mathcal{F}(p_i)$ together with features $\mathcal{F}(\mathbb{T}_i)$ to become features of the target class, adding training inputs $([\mathbb{T}_i, \mathbb{T}_j] \oplus \boldsymbol{x}_{p_i}, y_V)$ forces $\mathcal{F}(p_i) \cup \mathcal{F}(\mathbb{T}_i) \cup \mathcal{F}(\mathbb{T}_j)$ to become features of the victim class. The different labels of samples $\mathbb{T}_i \oplus \boldsymbol{x}_{p_i}$ and $[\mathbb{T}_i, \mathbb{T}_j] \oplus \boldsymbol{x}_V^{p_i}$ enable the model to learn new behaviors. As such, further stamping any other partition triggers to $[\mathbb{T}_i, \mathbb{T}_j] \oplus \boldsymbol{x}_V^{p_i}$ yields the same classification result, which is the victim class, according to the maximum trigger hypothesis.

Hypothesis 2 *Our training method is sufficient to achieve precise focusing, meaning that only $\mathbb{T}_i \oplus \boldsymbol{x}_V^{p_i}$ can yield the target label and stamping any other trigger or trigger combinations yields the victim class label.*

We can prove the hypothesis assuming the maximum subset hypothesis is correct. We focus on proving that an arbitrary non-empty set of triggers $\mathcal{T} \neq \{\mathbb{T}_i\}$ must yield the victim class label for $\boldsymbol{x}_V^{p_i}$. There are two possible cases. One is when $\mathbb{T}_i \notin \mathcal{T}$ and the other is when $\mathbb{T}_i \in \mathcal{T}$. In case one, without losing generality, assume $\mathcal{T} = \{\mathbb{T}_{t_1}, ..., \mathbb{T}_{t_k}\}$ with $0 < k < (n-1)$



(a) STRIP - CIFAR-10 (b) STRIP - ImageNet (c) Spectral Signatures - CIFAR-10 (d) Spectral Signatures - ImageNet

Figure 8. Evaluation against STRIP and Spectral Signatures on CIFAR-10 and ImageNet datasets.

and t_1, ..., t_k not equal to i. As such, $\{\mathbb{T}_{t_1}\}$, ..., $\{\mathbb{T}_{t_k}\}$ are the maximum subsets of $\mathcal{T}$ regarding p_i. They have a consistent label y_V. As such, $\overline{\mathcal{M}}_{\bar{\theta}}(\mathcal{T} \oplus \boldsymbol{x}_V^{p_i}) = y_V$ based on the maximum subset Hypothesis (1).

In case two, without losing generality, assume $\mathcal{T} = \{\mathbb{T}_i, \mathbb{T}_{t_1}, ..., \mathbb{T}_{t_k}\}$ with $0 < k < (n-1)$ and t_1, ..., t_k not equal to i. As such, $\{\mathbb{T}_i, \mathbb{T}_{t_1}\}$, ..., $\{\mathbb{T}_i, \mathbb{T}_{t_k}\}$ are the maximum subsets of $\mathcal{T}$ regarding p_i. They have a consistent label y_V. As such, $\overline{\mathcal{M}}_{\bar{\theta}}(\mathcal{T} \oplus \boldsymbol{x}_V^{p_i}) = y_V$. The hypothesis is hence proved. □

E. Evaluation against Meta-classifiers

Table 7. Evaluation on ULP and MNTD.

# Partitions	2	3	4	5	6	7	Accuracy
ULP	0	1	0	0	0	0	16.7%
MNTD	1	0	1	1	0	0	50.0%

In this section, we evaluate LOTUS against backdoor detection methods based on meta-classifier, i.e., MNTD [80] and ULP [29]. Both methods aim to train a binary classifier from a large number of benign and poisoned models for backdoor scanning. They generate a set of input patterns and feeds them to the models whose output logits are then used to train the classifier. During training, the patterns and the binary classifier are optimized together in order to distinguish benign and poisoned models in the training set. During inference, these patterns are fed to the given model whose output logits are used by the binary classifier to decide whether the model is poisoned.

As MNTD and ULP require a large number of models for training, we adopt the TDC dataset[2], which consists of 125 benign models and 125 poisoned models trained on CIFAR-10 using WRN. We are able to train a MNTD and ULP classifier with over 90% training accuracy. To evaluate LOTUS against both meta-classifiers, we generate 6 poisoned models using the same dataset and structure, with 2-7 partitions. Table 7 presents the detection results by MNTD and ULP, where 0 denotes benign and 1 poisoned. Observe that the detection accuracy of ULP is only 16.7%, and MNTD 50.0%, showing they are unable to detect LOTUS.

F. Evaluation Against Testing-time Sample Detection

We conduct experiments to study the evasiveness of LOTUS against 2 testing-time sample detection techniques, STRIP [18] and Spectral Signitures [62] on a ResNet18 model trained on CIFAR-10 using K-means implicit partitioning and a VGG16 model trained on RImageNet using secondary labeling. Specifically, for each model, we randomly select 400 clean and 400 poisoned samples and assess their distributions according to different baselines.

STRIP [18] examines the entropy of the resulting predictions to identify the presence of a backdoor trigger by superimposing an input with a set of clean samples. The overlapping distributions of the normalized entropy for clean and poisoned data in Figure 8a and 8b indicate that LOTUS evades the detection of STRIP. The reason is that super-imposing breaks condition of correct partitions to trigger the backdoor. On the other hand, Spectral Signatures [62] identifies backdoor attacks by detecting outliers in feature covariance spectra through singular value decomposition. Figure 8c and 8d visualizes the distributions of outlier scores for clean and poisoned samples, where they highly overlap, meaning Spectral Signatures is unable to detect LOTUS's poisoned samples. This is because LOTUS learns a complex mapping from partition secrets to trigger injection, making the internal values similar to that of complex benign features.

[2]https://trojandetection.ai/

G. Extension to Universal Attacks

We extend LOTUS to universal attack by partitioning samples based on their classes. Using ResNet18 as an example model, we divide CIFAR-10 samples into 5 partitions (2 classes for each partition) and apply Trigger Focusing for target label 0. We achieve 94.14% BA, 95.99% ASR and 5.94% ASR-other. Furthermore, the attacked model successfully evades the detection by NC, with an anomaly index of 1.614, which is below the established outlier threshold of 2.

H. Comparison with Additional Backdoor Attacks

We reproduce another two novel attack clean-label [64] and adaptive blend [48] on CIFAR-10 and ResNet-18 with over 94% BA and 99% ASR. NC can detect both of them and Fine-pruning can reduce their ASRs to <10%. However, NC and Fine-pruning cannot detect/mitigate LOTUS according to the results in Section 5.3 and Section 5.4, which indicates LOTUS outperforms these attacks regarding both evasiveness and resilience.

I. Evaluation against Additional Backdoor Mitigation Methods

We evaluate LOTUS on the two additional backdoor mitigation methods, i.e., I-BAU [15] and ARGD [14], using CIFAR-10 and ResNet-18. We create 4 victim paritions for LOTUS and assume the defender has access to 5% of the training data. Results are shown in Table 8, where we observe they are effective against the novel attack WaNet (ASR<10%), but fail to sufficiently defend LOTUS (ASR>30%). This is due to LOTUS's unique backdoor mechanism that relies on benign partitions, posing a great challenge for these mitigation methods.

Table 8. Evaluation against additional defense

Defense	LOTUS		**WaNet**	
	BA	ASR	BA	ASR
No Defense	91.54%	93.80%	91.22%	98.57%
I-BAU	88.95%	38.60%	88.84%	9.21%
ARGD	88.18%	31.30%	89.92%	1.39%

J. Discussion about the ASR of LOTUS

LOTUS achieves around 93% ASR across various models and datasets (Table 1), slightly lower than state-of-the-art attacks, e.g., Ftrojan [13] and WaNet [46]. However, LOTUS shows greater resilience to defenses. For instance, novel defense ANP [72] only reduces LOTUS's ASR to 35%, compared to under 3% for Ftrojan and WaNet, as shown in Table 9. We also explore a variant of LOTUS (LOTUS-rel.) by slightly reducing the weight on the dynamic loss (Equation 3). LOTUS-rel. increases ASR to over 98%, at a slight cost to resilience, reflecting a trade-off between ASR and resilience.

Table 9. Discussion about the ASR

Attack	**No Defense**		**ANP**	
	BA	ASR	BA	ASR
FTrojan	91.53%	99.95%	88.64%	2.09%
WaNet	91.22%	98.57%	89.07%	0.54%
LOTUS	91.54%	93.80%	88.14%	34.90%
LOTUS-rel.	91.76%	98.20%	90.10%	27.97%

K. Adaptive Defense

In this section, we assess the performance of LOTUS under various adaptive defense scenarios. We operate under the assumption that defenders have knowledge about the victim class and the number of partitions implemented by LOTUS. With this information, they can independently create partitions and utilize existing defense strategies to reverse-engineer a trigger for each partition and conduct detection. To illustrate, we conduct experiments using the ResNet18 model on CIFAR-10 attacked by LOTUS, employing four partitions generated via the implicit partitioning using K-means. We leverage NC [67] as a typical baseline for detection against LOTUS on partitioned samples. A model with anomaly index exceeding 2, as produced by NC,

Table 10. Evaluation on adaptive defense.

Partition Method	Par. 0	Par. 1	Par. 2	Par. 3	MO
GT	2.587	1.985	2.366	2.841	100%
Inputs	1.178	0858	0.691	0.832	47%
Internels	1.358	0.699	1.387	0.905	67%

Table 11. Evaluation on other partitions.

Metrics	Half 0 + half 1	Half 2 + half 3	Equal 4	Random
NC index	1.316	0.823	1.041	1.019
MO	50%	50%	25%	27%

is considered backdoored. Table 10 displays the outcomes for different partitioning techniques. The first column denotes the partitioning methods, with "GT" implying that the defender possesses precise knowledge of the partitions, "Inputs" suggesting that the defender employs K-means to generate partitions from input samples, and "Internals" meaning the defender utilizes K-means to partition based on internal feature representations. Subsequent columns represent NC anomaly indexes generated on various partitions. The last column, "MO" quantifies the maximum overlap between generated partitions and one of the ground-truth partitions. We observe that if defenders possess knowledge of the ground-truth partitions, they have a higher likelihood of detecting LOTUS by using samples from a specific partition. However, it is often impractical for defenders to have such knowledge. Scenarios where defenders generate partitions from input or feature data are more realistic. However, even when defenders employ the same partitioning algorithm on input samples and feature representations, they face difficulties in detecting LOTUS. This is because LOTUS leverages a surrogate model to learn partitioning principles, rather than directly using K-means results. Consequently, their partitioning outcomes differ, with an MO of only 67%. Table 11 presents additional results, testing detection scenarios where samples consist of an equal mix from two partitions (Half 0 + half 1), an equal mix from all four partitions (Equal 4), or random selections from partitions. In all instances, defenders find it challenging to detect LOTUS. In conclusion, even when defenders possess prior knowledge of LOTUS, detecting the backdoor remains challenging due to the complexity of its sub-partitioning approach.

L. Ablation Study

In this section, we conduct a series of ablation studies of LOTUS on different settings and hyper-parameters.

Table 12. Results w/ and w/o surrogate models.

Method	BA	ASR	ASR-other
K-means	94.36%	84.40%	19.01% $\pm$ 39.24%
K-means + surrogate	94.71%	94.30%	4.39% $\pm$ 17.08%
GMM	94.78%	86.38%	20.51% $\pm$ 40.38%
GMM + surrogate	94.59%	90.70%	4.80% $\pm$ 21.38%

L.1. Necessity of Using Surrogate Models

We verify the necessity of training a surrogate model for implicit sub-partitioning to attain high attack effectiveness with LOTUS. To validate this, we conducted experiments utilizing the ResNet18 model on CIFAR-10 and compared the attack performance achieved through traditional partitioning methods, specifically K-means and GMM, with the performance achieved when training a surrogate model for partitioning. The outcomes are presented in Table 12.

Notably, when we omit the surrogate model, the ASRs experience a noteworthy reduction, ranging from 4% to 10%. Additionally, ASRs-other exhibit an increase. This decline in performance is due to the difficulty of the attacked model to effectively learn the traditional partitioning schemes. This also highlights the necessity of training a surrogate model for sub-partitioning to achieve good attack performance.

L.2. Effect of Different Number of Partitions

We evaluate the attack's effectiveness by generating different numbers of partitions. The experiment is conducted on ResNet18 trained on CIFAR-10 using implicit sub-partitioning. The results are shown in Figure 9, with the x-axis representing the number of partitions and the y-axis indicating the percentage values for different metrics. These metrics include BA (Backdoor

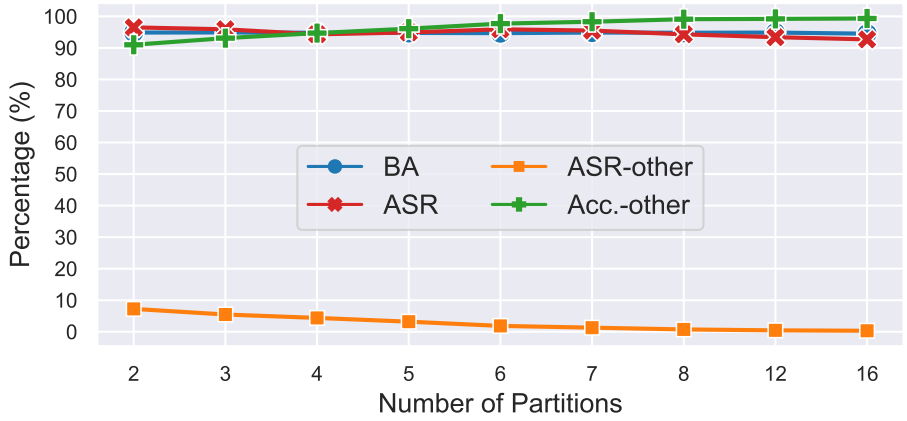


Figure 9. Attack effectiveness of different number of partitions

Accuracy), ASR (Attack Success Rate), ASR-other (Attack Success Rate of images assigned incorrect triggers), and Acc.-other (Accuracy for images with incorrect triggers).

It is noteworthy that BAs and ASRs consistently remain at high levels, showcasing the attack's effectiveness across various partition numbers, even when the number of partitions is as high as 12 or 16. Additionally, ASR-other values are low, while Acc.-others values are high, indicating that images marked with incorrect triggers tend to be predicted as belonging to their source labels. This observation further indicates that LOTUS achieves good performances across various partition counts.

Table 13. Results on different number of samples per partition.

# Samples per Partition	BA	ASR	ASR-other
1000 (default)	94.74%	93.90%	5.65%
500	93.94%	90.70%	7.09%
250	94.45%	91.20%	5.77%
100	92.00%	81.10%	9.46%
50	89.46%	79.40%	8.64%

L.3. Effect of Different Number of Samples per Partition

We explore the influence of different sample counts per partition, a crucial factor when employing sub-partitioning within the victim class. First, unevenly distributed samples resulting from partitioning could introduce fairness issues in learning. To mitigate this, we analyze the post-partitioning sample counts and strive to maintain a roughly balanced distribution, although sample imbalance rarely occurs (as discussed in Section 4.1).

Another potential challenge arises when the number of samples per partition is limited, potentially affecting trigger focusing due to the requisite knowledge of sub-partitioning secrets. To examine this effect, we conducted experiments using ResNet18 models on the CIFAR-10 dataset, generating 4 partitions within victim class 0. The results are displayed in Table 13, with the first column representing the number of samples per partition, followed by benign accuracy (BA), attack success rate (ASR), and attack success rate for non-targeted triggers (ASR-other). We compare LOTUS's performance with the default setting of 1000 samples per partition and other configurations. It is notable that as the number of samples per partition decreases, both BA and ASR exhibit declines, while ASR-other experiences an increase. These results indicate that LOTUS's performance degrades with fewer samples per partition. However, it's worth highlighting that even with just 250 samples, an ASR of over 91% can be achieved. This suggests that LOTUS generally requires around 1000 samples per victim class for good performance, which is often practical across various datasets.

L.4. Effect of Different Victim-Target Class pairs

In the previous experiments, we use the first class as the victim and the last class as the target as the default setting. Here, we randomly select a few victim-target class pairs to study how they affect the performance of LOTUS. We conduct an experiment on ResNet18 on CIFAR-10 and use implicit sub-partitioning to generate 4 partitions. Table 14 shows the results. Observe that LOTUS consistently has a high ASR with different victim-target class pairs, and the ASR-other values are relatively low, delineating the generalizability of LOTUS to different class pairs.

Table 14. Results on different victim-target class pairs.

Victim	Target	BA	ASR	ASR-other
0	9	95.01%	91.40%	4.65% ± 16.06%
4	5	94.94%	93.10%	3.64% ± 18.74%
1	6	95.13%	95.70%	5.44% ± 22.67%
7	4	95.03%	93.40%	5.89% ± 23.55%
3	2	95.21%	89.60%	6.38% ± 24.44%

Table 15. Results on different trigger patterns.

Trigger Pattern	BA	ASR	ASR-other
Color Patch	95.01%	91.40%	4.65% ± 16.06%
Logo	94.89%	91.40%	4.63% ± 16.00%
Instagram Filter	94.40%	89.10%	6.90% ± 21.24%

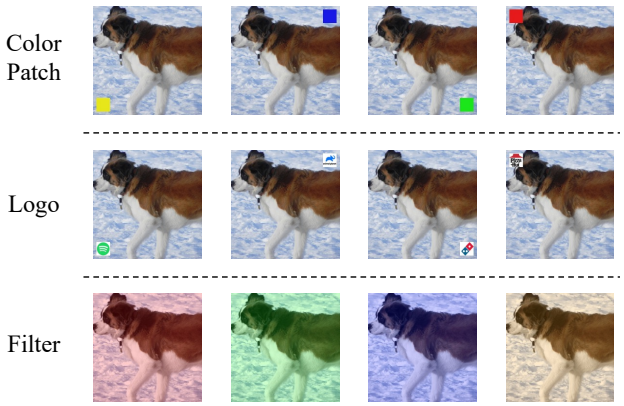


Figure 10. Different trigger patterns. The first row shows the color patch triggers, the second row logo triggers and the third row Instagram filter triggers.

L.5. Effect of Different Triggers

We use solid polygon patches as triggers in our previous experiments. We study two other types of triggers, logos and Instagram filters. Figure 10 shows different trigger patterns. The first row shows the color patch triggers, the second row the logo triggers (downloaded form the Internet), and the third row the Instagram filter triggers. We use a ResNet18 model on CIFAR-10 with 4 partitions for the study. Table 15 presents the results. All three studied cases have high BAs, ASRs, and low ASRs-other. Due to the overlapping of triggers during trigger focusing, there is a slight degradation of the attack effectiveness on Instagram filter, with 2% ASR degradation and 2% ASR-other increase. The Instagram filter cases perturb the entire image and hence slightly degrade the performance of trigger focusing. Overall, LOTUS is effective with different types of triggers.

Table 16. Results on different patch sizes.

Patch Size	BA	ASR	ASR-other
3×3	94.80%	93.90%	4.32%
6×6	94.89%	94.30%	4.39%
10×10	94.40%	94.40%	5.09%

L.6. Effect of Different Patch Trigger Sizes

We study the effect of different trigger sizes using solid patches as the trigger. We conduct experiments on ResNet18 model on CIFAR-10 with 4 secret partitions. We evaluate patch sizes of 3×3, 6×6 and 10×10. Results are shown in Table 16, where the first column denotes the trigger sizes with the subsequent columns illustrating LOTUS's performance, i.e., BA, ASR and ASR-other. Observe that LOTUS is generally effective using multiple trigger sizes, with high benign accuracy, ASR and low ASR-other.

Table 17. Results on different pre-trained encoders.

Encoder	BA	ASR	ASR-other
VGG	94.71%	94.30%	4.39% ± 17.08%
AlexNet	95.10%	92.70%	4.59% ± 20.92%
SqueezeNet	94.75%	91.60%	4.26% ± 20.21%

L.7. Effect of Different Pre-trained Encoders

In the previous experiments, we leverage the pre-trained encoders of VGG [84] to extract features of victim samples. We study other two encoders of different structures. We conduct experiments on a ResNet18 model on CIFAR-10 with 4 implicit partitions. Table 17 shows the results of using different pre-trained encoders. Observe that LOTUS achieves a consistent good performance through out all encoders.

M. Evaluation Against GradCAM

The GradCAM [53] method is commonly used to visualize the important regions of inputs through gradient propagation. In this study, we evaluate LOTUS using GradCAM and compare the results with those of BadNets [19] and Dynamic [51] backdoors. Figure 11 displays the results, which are organized into four groups of images representing the GradCAM visualizations for the clean model, the BadNets attacked model, the Dynamic attacked model, and the LOTUS attacked model. In each group of images, the first row shows the poisoned images (clean images for the clean model), while the second row shows the GradCAM visualizations. The reddish regions represent important areas, while the bluish regions represent less important parts.

Our observations revealed that for both BadNets and Dynamic backdoors, the important regions are located at the trigger positions, indicating that their triggers have notable features regarding the gradients. However, for LOTUS's poisoned images, the important regions are more similar to those of the clean models. This further validates the evasiveness of LOTUS and explains why it is difficult to be inverted.

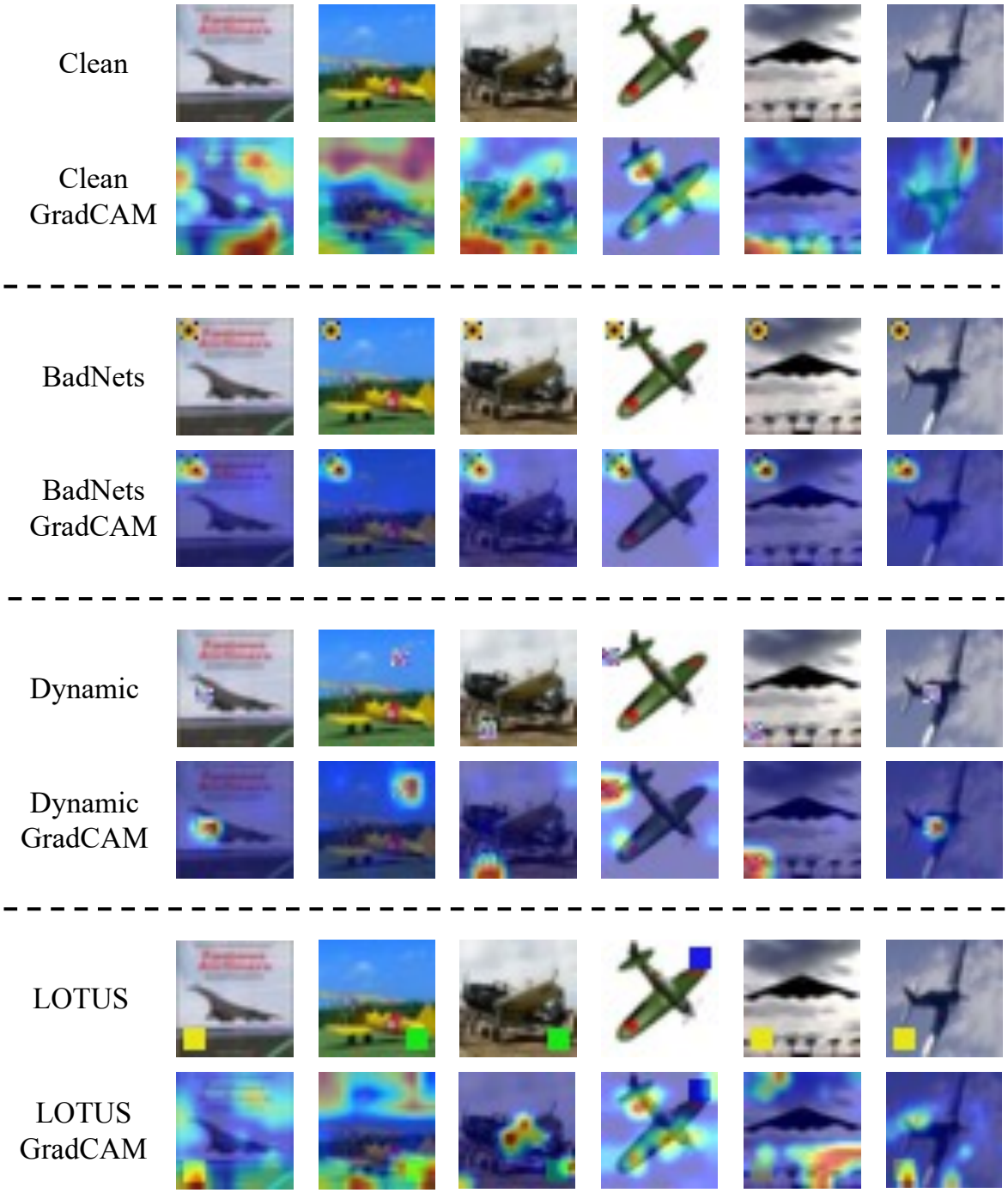


Figure 11. Important region visualizations via GradCAM.