A Understanding the Fine-tuning Process of PLMs on Poisoned Datasets

In this section, we show our empirical observations obtained from fine-tuning PLMs on poisoned datasets. Specifically, we demonstrate that the backdoor triggers are easier to learn from the lower layers than the features corresponding to the main task. This observation plays a pivotal role in designing and understanding our defense algorithm. In our experiment, we focus on the SST-2 dataset [30] and consider the widely adopted word-level backdoor trigger and the more stealthy style-level trigger. For the word-level trigger, we follow the approach in prior work [25] and adopt the meaningless word "bb" as the trigger to minimize its impact on the original text's semantic meaning. For the style trigger, we follow previous work [10] and select the "Bible style" as the backdoor style. For both attacks, we set a poisoning rate at 5% and conduct experiments on the RoBERTa$_{\text{BASE}}$ model [31], using a batch size of 32 and a learning rate of 2e-5, in conjunction with the Adam optimizer [32]. To understand the information in different layers of PLMs, we draw inspiration from classifier probing studies [33, 34] and train a compact classifier (one RoBERTa transformer layer topped with a fully connected layer) using representations from various layers of the RoBERTa model. Specifically, we freeze the RoBERTa model parameters and train only the probing classifier.

In Figure 6, we present the training loss curve of the word-level trigger, which utilizes a probing classifier constructed using features extracted from twelve different layers of the RoBERTa model. A critical observation highlights that in the initial layers (1-4), the probing classifier overfits the poisoned samples early in the training phase (around 500 steps). However, it underperforms the original task. This can be attributed to the initial layers primarily capturing surface-level features, including phrase-level and syntactic-level features, which are insufficient for the primary task. Subsequently, in Figure 7, we delve deeper into the visualization of the probing classifier's CLS token embeddings. A notable demarcation can be observed between the embeddings for poisoned and clean samples across all layers. However, the distinction between positive and negative sample embeddings becomes less discernible in the lower layers. We found a similar trend for the style-level trigger, as we showed the learning dynamic in Figure 8 and embedding visualization in Figure 9.

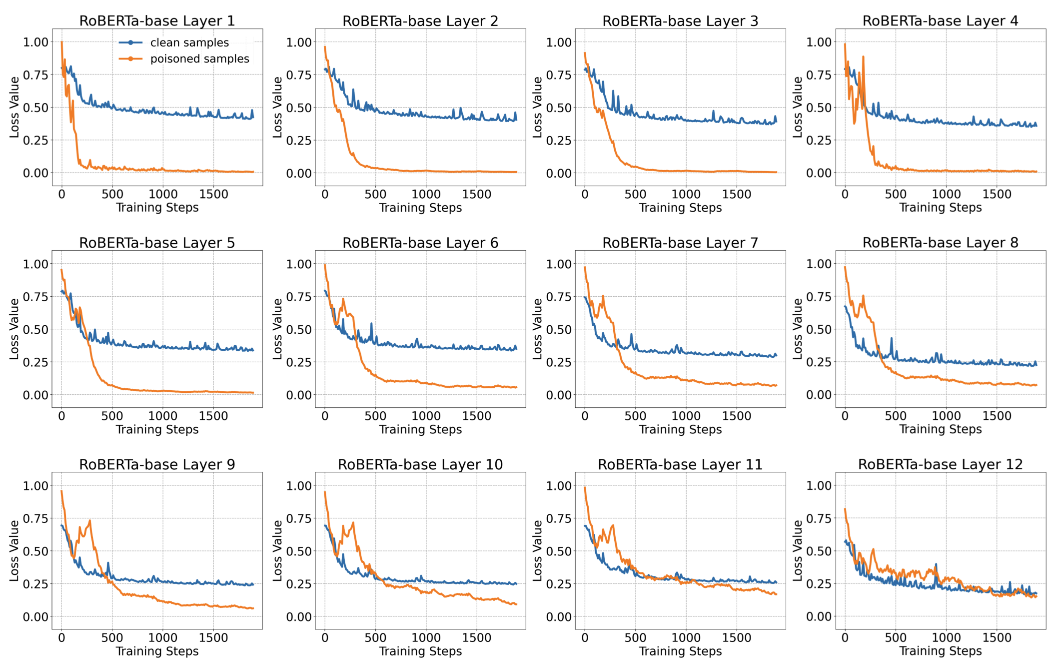


Figure 6: Learning dynamic for Word-level Trigger

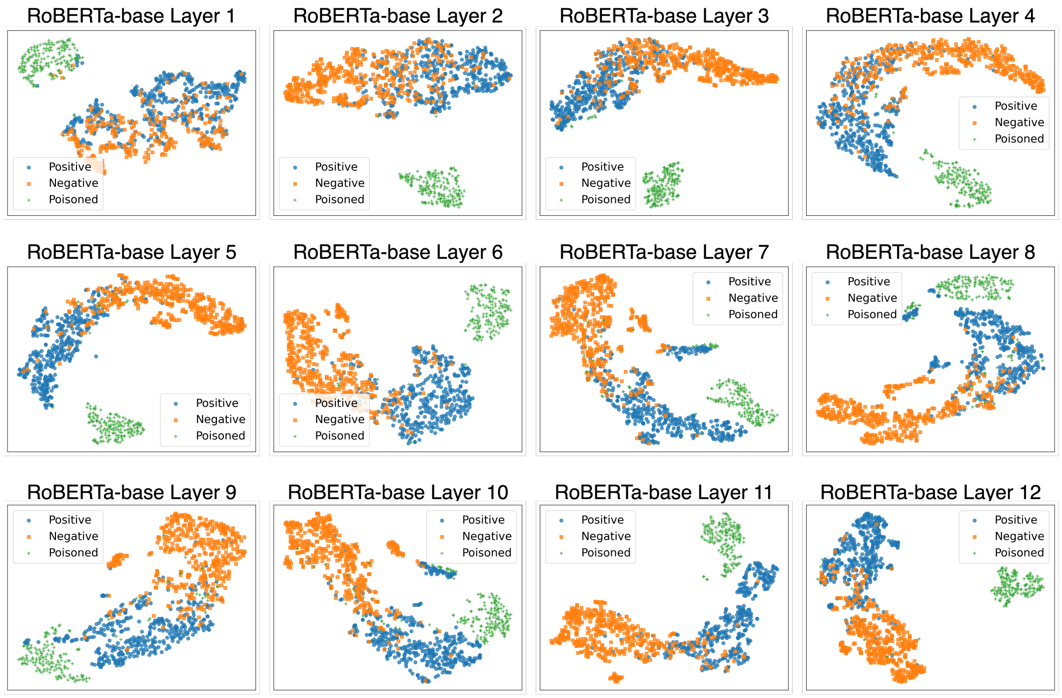


Figure 7: Embedding Visualization for Word-level Trigger

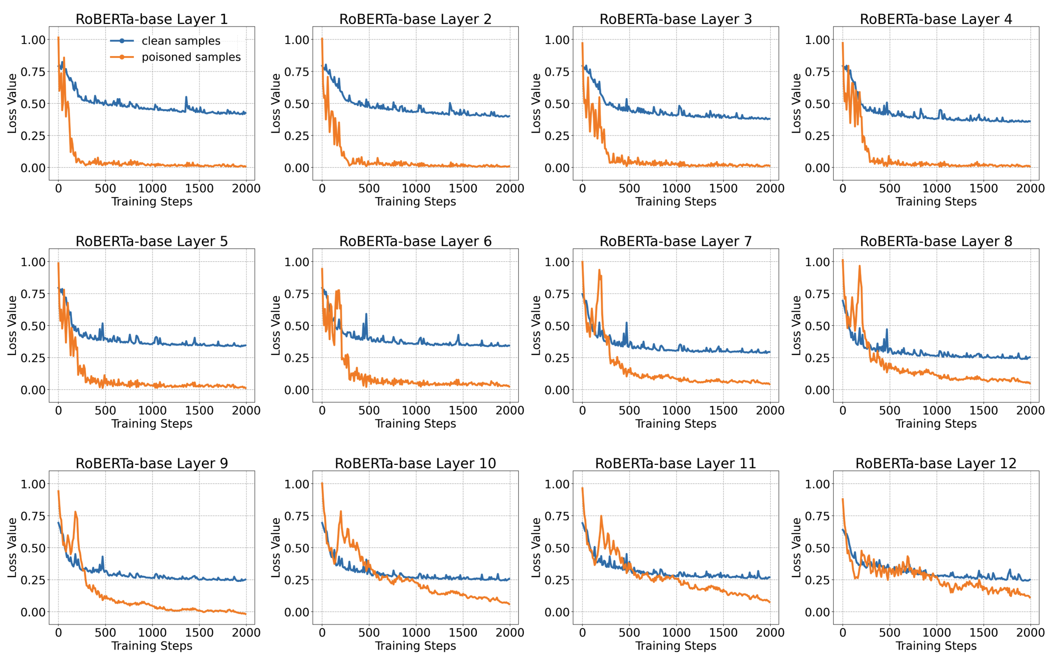


Figure 8: Learning dynamic for Style-level Trigger

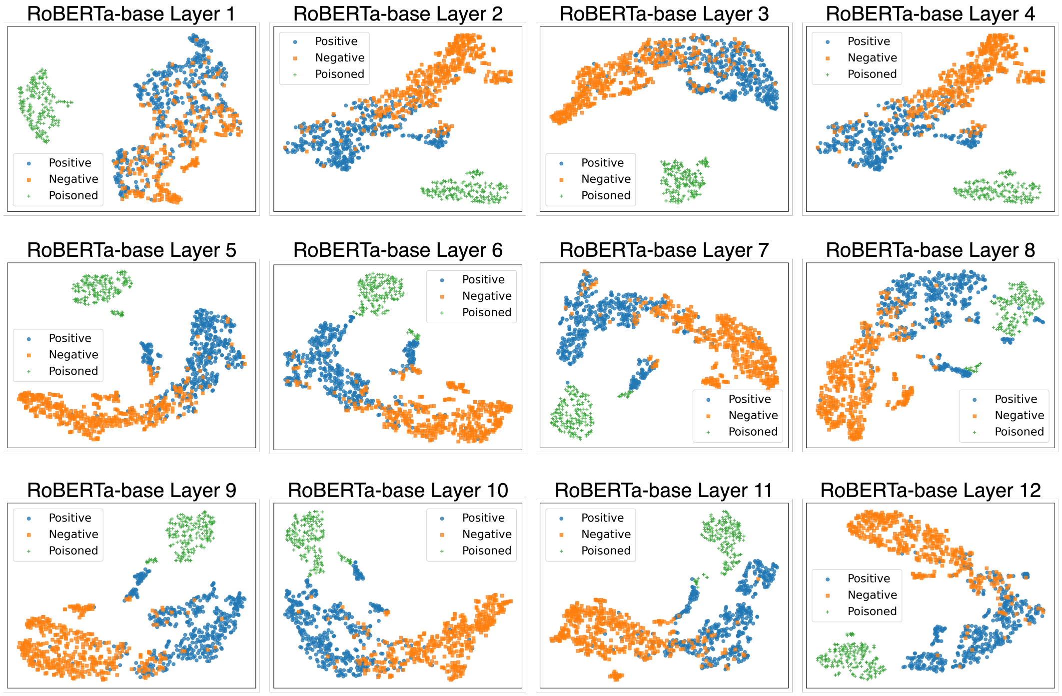


Figure 9: Embedding Visualization for Style-level Trigger

B More on Defense Results

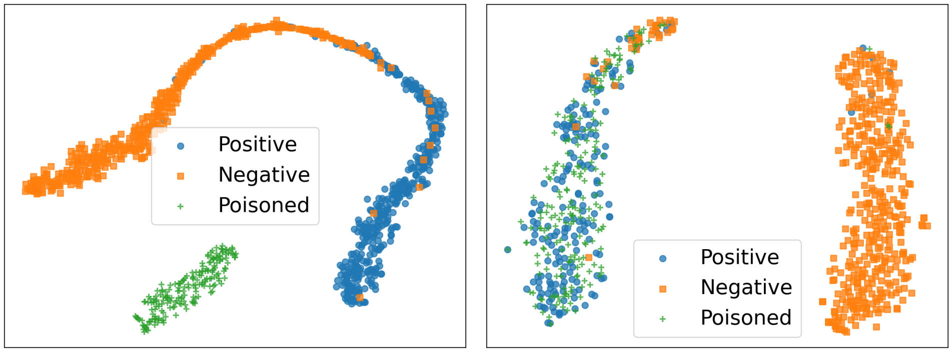
Table 6: Performance comparison with other defense methods on IMDB dataset

Defence Method	AddWord		AddSent		StyleBKD		SynBKD	
	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)
No defense	93.88 ±0.76	100.00 ±0.00	93.68 ±0.72	100.00 ±0.00	93.92 ±0.68	99.52 ±0.15	93.84 ±0.72	99.53 ±0.20
BKI	93.32 ±0.87	87.27 ±2.90	92.84 ±0.90	98.65 ±1.10	93.10 ±0.85	99.02 ±0.30	93.00 ±0.90	99.35 ±0.25
ONION	88.32 ±0.94	32.32 ±3.02	89.76 ±0.92	89.04 ±3.70	88.32 ±0.96	95.58 ±0.38	88.80 ±0.95	99.65 ±0.10
RAP	93.10 ±0.84	85.62 ±3.58	92.70 ±0.88	91.20 ±3.45	92.96 ±0.86	76.90 ±3.80	92.70 ±0.90	78.80 ±3.70
STRIP	93.74 ±0.78	97.90 ±3.20	93.70 ±0.80	100.00 ±1.20	93.50 ±0.82	78.70 ±1.50	93.60 ±0.78	88.90 ±1.10
MF	92.80 ±0.86	21.30 ±3.50	92.60 ±0.90	36.00 ±2.50	92.80 ±0.85	65.80 ±2.80	92.90 ±0.88	76.50 ±2.20
Our Method	**93.72** ±0.84	**5.60** ±2.04	**92.72** ±0.88	**6.56** ±2.23	**93.12** ±0.89	**19.36** ±2.90	**93.20** ±0.93	**22.70** ±2.80

Table 7: Performance comparison with other defense methods on OLID dataset

Defence Method	AddWord		AddSent		StyleBKD		SynBKD	
	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)
No defense	85.23 ±0.68	99.83 ±0.25	85.00 ±0.67	100.00 ±0.00	84.88 ±0.71	99.24 ±0.39	85.23 ±0.67	100.00 ±0.00
BKI	84.76 ±0.89	90.23 ±2.67	84.88 ±0.84	100.00 ±0.00	83.23 ±0.98	98.34 ±0.42	83.72 ±0.95	99.61 ±0.25
ONION	84.41 ±0.88	58.10 ±2.34	85.11 ±0.82	100.00 ±0.00	85.11 ±0.86	99.63 ±0.31	84.53 ±0.92	99.39 ±0.30
RAP	83.93 ±0.90	87.18 ±3.11	83.72 ±0.94	99.44 ±0.35	83.54 ±0.97	95.23 ±1.93	83.91 ±0.89	94.45 ±1.95
STRIP	85.00 ±0.76	100.00 ±0.00	83.27 ±0.86	99.25 ±0.30	84.65 ±0.91	88.81 ±0.25	83.98 ±0.93	79.84 ±0.20
MF	81.97 ±0.93	21.24 ±2.92	81.86 ±0.97	68.92 ±2.79	82.09 ±0.98	68.42 ±3.10	82.89 ±0.92	58.52 ±3.00
Our Method	**82.79** ±0.85	**11.45** ±3.17	**83.37** ±0.82	**4.83** ±2.04	**83.95** ±0.90	**29.18** ±2.92	**83.02** ±0.93	**28.40** ±2.90

In this section, we delve deeper into the comparison between our method and several other backdoor defense strategies, maintaining the same conditions as outlined in Section 5. Particularly, Table 6 shows our honeypot technique against others on the $\text{RoBERTa}_{\text{BASE}}$ with the IMDB dataset. Additionally, results using the OLID dataset are presented in Table 7. In the case of the IMDB dataset, our method consistently achieves the lowest ASR across all four attack methods, displaying a robust defense technique even under varied adversarial conditions. For example, considerting the AddWord and AddSent attacks, our ASR is below 10%, which is a considerable improvement over other methods. In StyleBKD and SynBKD, our ASR stays below 23%, still outperforming the competing



(a) No Defense (b) Our Method

Figure 10: Embedding Visualization for Victim Model and Protected Model

methods by a wide margin. Similarly, for the OLID dataset, our method demonstrated excellent performance, surpassing all other defense methods in terms of ASR. Furthermore, our method still achieves competitive ACC results on the original tasks. In Figure 10, we exhibit the t-SNE visualizations derived from the CLS token embeddings of the final transfer layer of the RoBERTa model. As shown in Figure 10 (a), we observe that the no-defense model clearly recognizes the poisoned samples. Instead, in Figure 10 (b), the model overlooks the backdoor trigger and successfully predicts positive samples with embedded backdoor words as the positive class.

C Understanding the Honeypot Defense Training Process

In this section, we further illustrate more details about the honeypot defense training process. Specifically, we focus on the dynamic change of the training weight for poisoned and clean samples. As we mentioned in Section 4, we propose employing a weighted cross-entropy loss ($\mathcal{L}_{WCE}$):

$$\mathcal{L}_{WCE}(f_T(x), y) = \sigma(W(x) - c) \cdot \mathcal{L}_{CE}(f_T(x), y), \quad \text{where} \tag{5}$$

$$W(x) = \frac{\mathcal{L}_{CE}(f_H(x), y)}{\bar{\mathcal{L}}_{CE}(f_T(x), y))}, \tag{6}$$

$f_H(x)$ and $f_T(x)$ represent the softmax outputs of the honeypot and task classifiers, respectively. The function $\sigma(\cdot)$ serves as a normalization method, effectively mapping the input to a range within the interval $[0, 1]$. The c is a threshold value for the normalization.

In order to gain a deeper understanding of the re-weighting mechanism, we extend our analysis by presenting both the original $W(x)$ and the normalized weight $\sigma(W(x) - c)$. We conducted the experiment using the SST2 dataset, with a word-level trigger, a poisoning rate set at 5%, and a batch size of 32. Figure 11 illustrates the $W(x)$ value for both the poisoned and clean samples at each stage of training. Specifically, we computed the $W(x)$ for each mini-batch and then calculated the average $W(x)$ value for both the poisoned and clean samples. As depicted in the figure, during the warm-up phase, the $W(x)$ for clean and poisoned samples diverged early in the training process. After 500 steps, the $W(x)$ for poisoned samples was noticeably lower than for clean samples. After the warm-up stage, given that $W(x)$ is higher for clean samples, the Cross-Entropy loss of clean samples in f_T diminishes more quickly than that of the poisoned samples. This subsequently increase $W(x)$ for clean samples as they possess a smaller $\bar{\mathcal{L}}_{CE}(f_T(x), y))$. This positive feedback mechanism ensures that the $W(x)$ for poisoned samples persistently remains significantly lower than for clean samples throughout the complete training process of f_T. As demonstrated in Figure 11, the $W(x)$ for the clean samples will continue to increase following the warm-up phase.

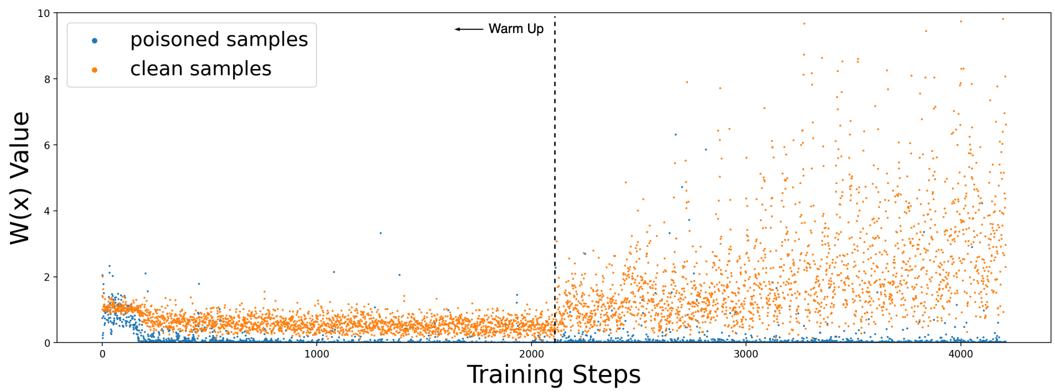


Figure 11: Visualization of W(x) during defense training process.

D More on Ablation Studies

D.1 Ablation Study on Honeypot Warm-Up

In the following section, we explore the influence of the preliminary warm-up steps in the honeypot method, which represent the number of optimizations that the honeypot branch requires to capture a backdoor attack. We applied our method against word-level attacks on RoBERTa$_{\mathrm{BASE}}$, and the obtained results are shown in Table 8. The analysis indicates that with a minimum count of warm-up steps, specifically below 200 for the SST-2 dataset, the honeypot is insufficiently prepared to capture the poisoned data. However, once the honeypot accrues a sufficient volume of poisoned data, surpassing 400 training steps across all datasets, the Attack Success Rate (ASR) can be mitigated to an acceptably low level, i.e., less than 10%. The results further prove that our honeypot can effectively capture backdoor information with a certain amount of optimization. In our main experiments, we set the number of warm-up steps equal to the steps in one epoch, thereby enabling our honeypot to reliably catch the poisoned data.

Table 8: Impact of Warm-Up steps

Dataset	SST-2					IMDB				
Warm-Up Steps	100	200	400	1000	2000	100	200	400	1000	2000
ACC (↑)	94.61	94.72	94.50	94.41	94.15	94.71	94.80	94.26	94.33	94.12
ASR (↓)	100.00	100.00	8.64	5.37	5.84	100.00	7.62	5.32	5.79	3.60

D.2 Ablation Study on Normalization Method

In this section, we use the SST2 dataset and word-level trigger to understand the impact of different normalization functions. As outlined in Section 4, our approach employs a normalization method to map the training loss weight $W(x)$ into the $[0, 1]$ interval. Within our experiments, we opted for the sign function as the normalization technique. However, we also explored two alternative normalization strategies – the sigmoid function and a cutoff ReLU function. For the latter, we assigned a value of 1 to any input exceeding 1. As depicted in Table 9, we conducted the experiments on RoBERTa$_{\mathrm{BASE}}$ using different normalization functions, we can observe that all normalization methods demonstrate decent performance in minimizing the ASR. Notably, we observe that the sign function yields the highest ACC on the original task while simultaneously achieving the lowest ASR.

Table 9: Impact of Normalization Method

Normalization	**AddWord**	
	ACC (↑)	ASR (↓)
No Defense	94.61$\pm$0.60	100.00$\pm$0.00
Sign	93.71$\pm$0.68	6.56$\pm$1.91
Sigmoid	93.22$\pm$0.53	6.83$\pm$2.01
Cutoff Relu	93.10$\pm$0.71	6.77$\pm$1.04

E Extend Honeypot to Computer Vision Tasks

While this paper primarily focuses on defending pretrained language models against backdoor attacks, we also explored the applicability of our proposed honeypot defense method within the computer vision domain [3, 4, 5]. In Section E.1, we illustrate the experimental settings. In Section E.2, we show the empirical findings. In Section E.3, we discuss the defense performance.

E.1 Settings

Suppose $D_{train} = (x_i, y_i)$ indicates a benign training dataset where $x_i \in \{0, ..., 255\}^{C \times W \times H}$ represents an input image with C channels and W width and H height, and y_i corresponds to the associated label. To generate a poisoned dataset, the adversary selects a small set of samples D_{sub} from the original dataset D_{train}, typically between 1-10%. The adversary then chooses a target misclassification class, y_t, and selects a backdoor trigger a and $a \in \{0, ..., 255\}^{C \times W \times H}$. For each instance (x_i, y_i) in D_{sub}, a poisoned example (x_i', y_i') is created, with x_i' being the embedded backdoor trigger of x_i and $y_i' = y_t$. The trigger embedding process can be formulated as follows,

$$x_i' = (1 - \lambda) \otimes x + \lambda \otimes a, \tag{7}$$

where $\lambda \in [0, 1]^{C \times W \times H}$ is a trigger visibility hyper-parameter and $\otimes$ specifies the element-wise product operation. The smaller the λ, the more invisible the trigger and the more stealthy. The resulting poisoned subset is denoted as $D'sub$. Finally, the adversary substitutes the original D_{sub} with D'_{sub} to produce $D_{poison} = (D_{train} - D_{sub}) \cup D'_{sub}$. By fine-tuning PLMs with the poisoned dataset, the model will learn a backdoor function that establishes a strong correlation between the trigger and the target label y_t. Consequently, adversaries can manipulate the model's predictions by adding the backdoor trigger to the inputs, causing instances containing the trigger pattern to be misclassified into the target class t.

In our experiment, we employed an ImageNet pretrained VGG-16 model as our base architecture and proceed with experiments using a manipulated CIFAR-10 dataset. The experiments involve the use of a 3 x 3 white square and a black line with a width of 3 pixels as backdoor triggers. The white square trigger is positioned at the bottom-right corner of the image, while the black line trigger is set at the bottom. We establish a poison rate of 5% and set $\lambda \in \{0, 0.2\}^{C \times\ W \times H}$ for two attacks. The values of λ corresponding to pixels situated within the trigger area are 0.2, while all others are set to 0.

E.2 Lower Layer Representations from VGG Provide Sufficient Backdoor Information

Drawing on our analysis presented in Section 3, we delve further into understanding the information encapsulated within various layers of a pretrained computer vision model. Inspired by previous classifier probing studies [33, 34], we train a compact classifier using representations derived from different layers of the VGG model. We ensure the VGG model parameters are frozen during this process and only train the probing classifier. In this context, we divided the VGG model into five sections based on the pooling layer operations (The five pooling layers are located at layers 2, 4, 7, 10, and 13). Subsequent to this, we integrate an adaptive pooling layer to reduce the features extracted from different layers to 7×7, ensuring that the flattened dimension does not exceed 8000. A fully connected layer with softmax activation is added as the final output. As depicted in Figure 13 and Figure 12, it is noticeable that the lower layers of the VGG model hold sufficient information for identifying the backdoor triggers. However, they do not contain enough information to effectively carry out the main tasks.

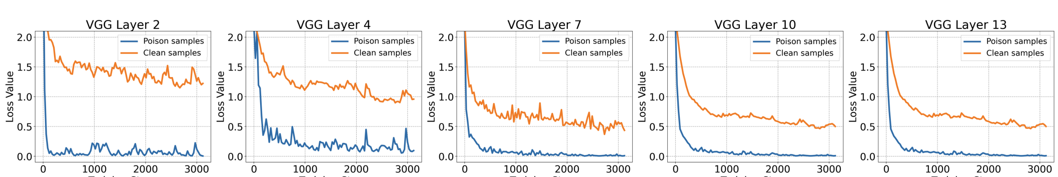


Figure 12: Learning Dynamic for White Square Trigger

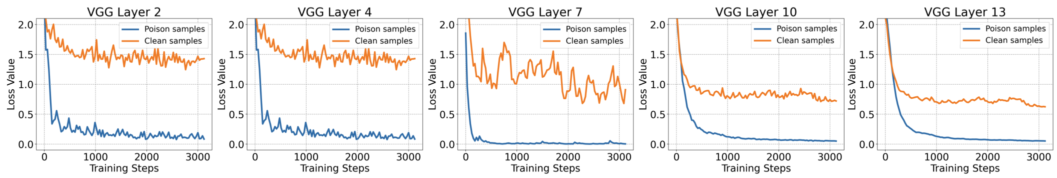


Figure 13: Learning Dynamic for Black Line Trigger

Table 10: Defense Performance on CIFAR10

Method	White Square		Black Line	
	ACC (↑)	ASR (↓)	ACC (↑)	ASR (↓)
No Defense	91.33±0.27	100.00±0.00	91.28±0.13	100.00±0.00
Our Method	92.20±0.43	8.81±1.09	92.23±0.37	10.81±1.83

E.3 Defense Results on CIFAR10

We implemented the honeypot as mentioned in Section 4 and built the honeypot module with the features from the first pooling layer. We followed previous sections and adopted the ASR and ACC metrics to measure the model's performance on the poisoned test set and clean test set, respectively. Specifically, we executed a fine-tuning process for a total of 10 epochs, incorporating an initial warmup epoch for the honeypot module. The learning rates for both the honeypot and the principal task are adjusted to a value of 1×10^{-3}. Additionally, we established the hyperparameter q for the GCE loss at 0.5, the time window size T was set to 100, and the threshold value c was fixed at 0.1. Each experimental setting was subjected to three independent runs and randomly chosen one class as the target class. These runs were also differentiated by employing distinct seed values. The results were then averaged, and the standard deviation was calculated to present a more comprehensive understanding of the performance variability. As the results are shown in Table 10, the proposed method successfully defends two backdoor attacks and reduces the ASR to lower than 10%. This indicates that the proposed method is valid for those simple vision backdoor triggers while having minimal impact on the original task. We plan to test the defense performance of more advanced backdoor triggers in our future work.

F Reproducibility

In an effort to ensure the reproducibility of our results, we have shared our test code along with the model checkpoint. This will allow peers in the research community to validate our findings. In the interest of complete transparency, we are also committed to releasing our training code in the future. This will provide a comprehensive understanding of our methodology and enable fellow researchers to extend and build upon our work. Our code can be found at `https://anonymous.4open.science/r/honeypot-backdoor-600E`.

G Limitations and Discussions

In this study, we introduce an innovative approach to backdoor defense in the context of fine-tuning pretrained language models. Due to the constraints in terms of time and resources, our evaluations were conducted using four prevalent backdoor attack methods and on three representative datasets. Despite the robustness and consistency demonstrated by our method, it is essential to remain vigilant to the emergence of new and potentially threatening attack methods and datasets, especially considering the rapid growth of this field. In addition, it's worth acknowledging that while unintended, some malicious users may exploit our method and deploy other strong backdoor attacks that may bypass our defense system.