LMSanitator: Defending Prompt-Tuning Against Task-Agnostic Backdoors

Chengkun Wei*¶, Wenlong Meng*¶, Zhikun Zhang*†‡‖, Min Chen‡, Minghu Zhao*,
Wenjing Fang§, Lei Wang§, Zihui Zhang*, Wenzhi Chen*‖
*Zhejiang University, †Stanford University, ‡CISPA Helmholtz Center for Information Security, §Ant Group
*{weichengkun, mengwl, Kinson_zhao, zhangzihui, chenwz}@zju.edu.cn, †zhikun@stanford.edu,
‡min.chen@cispa.de, §{bean.fwj, shensi.wl}@antgroup.com

***Abstract*—*Prompt-tuning* has emerged as an attractive paradigm for deploying large-scale language models due to its strong downstream task performance and efficient multitask serving ability. Despite its wide adoption, we empirically show that prompt-tuning is vulnerable to downstream task-agnostic backdoors, which reside in the pretrained models and can affect arbitrary downstream tasks. The state-of-the-art backdoor detection approaches cannot defend against task-agnostic backdoors since they hardly converge in reversing the backdoor triggers. To address this issue, we propose LMSanitator, a novel approach for detecting and removing task-agnostic backdoors on Transformer models. Instead of directly inverting the triggers, LMSanitator aims to invert the *predefined attack vectors* (pretrained models' output when the input is embedded with triggers) of the task-agnostic backdoors, which achieves much better convergence performance and backdoor detection accuracy. LMSanitator further leverages prompt-tuning's property of freezing the pretrained model to perform accurate and fast output monitoring and input purging during the inference phase. Extensive experiments on multiple language models and NLP tasks illustrate the effectiveness of LMSanitator. For instance, LMSanitator achieves 92.8% backdoor detection accuracy on 960 models and decreases the attack success rate to less than 1% in most scenarios.**[1]

I. Introduction

High-quality *language models* are critical for modern NLP tasks [17], [43], [28], yet their training requires substantial resources. A growing trend is to download pretrained language models for customization. *Fine-tuning* is a common paradigm to adapt *pretrained models* to downstream tasks. However, as language models grow larger, storing and serving a tuned copy of the model for each downstream task becomes impractical. To simultaneously achieve strong downstream task performance and efficient multitask serving ability, researchers proposed the *prompt-tuning* paradigm [39], [29], [34], [54], [38] as an alternative to fine-tuning. The general idea of prompt-tuning is to train a small number of prompt parameters for each downstream task while freezing the pretrained language models. As such, it allows the migration of pretrained models to downstream tasks without changing their parameters. Prompt-tuning has been proven to achieve comparable or even better performance than fine-tuning [38], which makes it attractive to individual researchers and small companies.

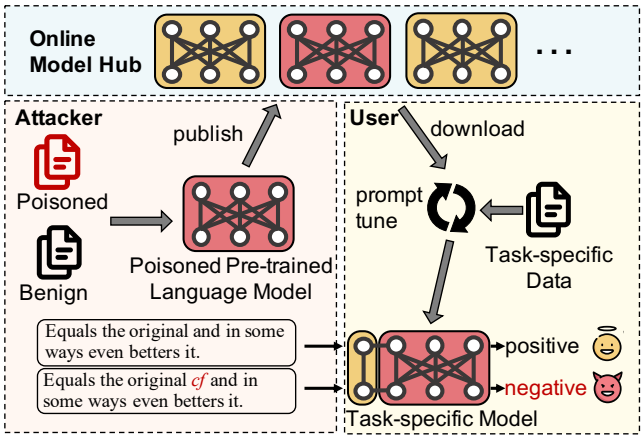


Fig. 1: Task-agnostic backdoors against prompt-tuning.

However, users who download the models from the public online hub, such as HuggingFace, might face security risks [61], [84], [76], [27], [10], [64], [80] due to a lack of security checks on these open-sourced language models. For instance, malicious entities might implant *backdoors* to the pretrained model and aim to affect the behaviors of the downstream tasks. The backdoored model functions normally on benign inputs while producing anomalous behavior on inputs containing specific attack patterns (or *triggers*). The most powerful backdoor attack against pretrained models is the *task-agnostic backdoor*, which can affect arbitrary downstream tasks. Figure 1 illustrates a typical threat of task-agnostic backdoors in the model supply chain. Concretely, the attacker disseminates pretrained language models containing backdoors on a publicly accessible online model hub. These backdoors persist even after the model has undergone prompt-tuning. Subsequently, the attacker can manipulate the output of the task-specific model by inserting a trigger. Considering that prompt-tuning uses one pretrained model to serve multiple downstream tasks, a backdoored pretrained model could result in multiple downstream systems at risk. Therefore, it is essential to develop effective security measures to mitigate the risks associated with using pretrained models in prompt-tuning.

To mitigate the security risks of backdoor attacks, re-

¶The first two authors made equal contribution.
‖Zhikun Zhang and Wenzhi Chen are corresponding authors.
[1]Code is available at https://github.com/meng-wenlong/LMSanitator.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23238
www.ndss-symposium.org

searchers proposed multiple backdoor detection approaches in the NLP field [71], [44], [2], of which RAP [44] is the state-of-the-art against task-specific backdoors. Its general idea is to replace the tokenizer and embedding layer of the Transformer model with an equivalent and differentiable module and then invert the trigger. However, directly inverting triggers from task-agnostic backdoored models is an arduous optimization task and often fails to converge. We observe that RAP's convex hull is small on task-agnostic backdoors and is surrounded by peaks that prevent the model from converging (see Figure 11b in Appendix B). LCP needs to add additional word encoding layers in front of the Transformer model, making the model deeper and harder to converge. Furthermore, existing backdoor removal methods, which aim to disable backdoors, require supplementary model updating and storage [37], [35]. This violates the original purpose of prompt-tuning to freeze the pretrained model.

Our Contributions. In this paper, we first comprehensively evaluate the security risks of prompt-tuning through task-agnostic backdoor attacks. We then propose LMSanitator, a new defense mechanism to detect task-agnostic backdoors on Transformer models and remove triggers during the inference phase. The role of LMSanitator is twofold: First, to help users determine whether a pretrained language model contains task-agnostic backdoors; Second, to protect downstream prompt-tuning models from backdoor interference. For instance, when a developer working on a downstream task model retrieves a potentially suspicious pretrained model from the Internet, they can utilize LMSanitator to ascertain the presence of task-agnostic backdoors within the model. If the result confirms the existence of backdoors, the developer has the option to either discard the compromised model in favor of a different one or continue using the tainted model while employing the trigger removal functionality of LMSanitator to supervise the input and filter out any triggers.

Instead of inverting precise trigger words, LMSanitator aims to invert exceptional output caused by task-agnostic backdoors (we define what kind of feature output is exceptional in Section IV-B). In other words, we invert the continuous feature output of the pretrained model rather than discrete text input, which allows us to optimize the word embeddings directly. LMSanitator randomly inserts trainable word embeddings to the input embeddings and updates these word embeddings to trigger task-agnostic backdoors. LMSanitator adds no extra layers and has a larger convex hull than RAP or LCP (see Figure 11c in Appendix B). After obtaining exceptional outputs, LMSanitator monitors the output of the language model during the inference phase. If the model output is similar to an inverted exceptional output, LMSanitator confirms that the input contains a trigger. LMSanitator only requires the defender to have some clean sentences, which is realistic in practice. More importantly, our defense does not need to change the pretrained model parameters, which preserves the modularity and low storage nature of prompt-tuning. The key contributions of this paper are summarized below.

- We are the first to investigate task-agnostic backdoor attacks against the state-of-the-art prompt-tuning models, including P-tuning and P-tuning v2. We empirically show that prompt-tuning is more vulnerable to backdoor attacks than fine-tuning on various tasks, such as sentence classification and named entity recognition (NER).
- To the best of our knowledge, LMSanitator is the first method to detect and remove task-agnostic backdoors without changing the model parameters, which maintains the modularity and storage cost of prompt-tuning. Moreover, LMSanitator only requires the defender to have some clean sentences and does not require any knowledge of the attacker, which is practical to implement.
- We evaluate LMSanitator on 3 types of task-agnostic backdoor attacks against a dozen of state-of-the-art language models and 8 downstream tasks. Within 960 models (half clean and half backdoored), LMSanitator gains a 92.8% backdoor detection accuracy. For all 252 backdoors embedded in 42 models, LMSanitator can find 239 of them, which achieves a 94.8% backdoor recall. LMSanitator can reduce the attack success rate (ASR) to 1% in most cases without changing the model parameters. We also test two models on HuggingFace published by NeuBA [84], each containing six backdoors. LMSanitator can find 11 out of 12 backdoors. Our experiments demonstrate that LMSanitator is also robust to adaptive attacks.

II. Preliminaries

A. Language Models and Prompt-tuning

Language Models. Language models are widely used in a variety of real-world applications, such as sentiment analysis [12], [50], [65], neural translation [75], [1], and question-answering [15], [30]. Modern language models use Transformer [70] as their backbone and contain billions of parameters, e.g., the minimal version of Stanfold Alpaca [68] (open source alternative to OpenAI ChatGPT) contains 7 billion parameters. Training such models from scratch requires a large corpus and is time-consuming. To address this issue, researchers propose the fine-tuning paradigm [18], [55]. Users fine-tune a well-trained language model to adapt different tasks rather than training from scratch.

From Fine-tuning to Prompt-tuning. As language models become larger, storing and serving a tuned copy of them for each downstream task becomes resource-exhaustive. Prompt paradigm aims to address this issue. Its core idea is to append a well-designed prompt to the input sentence for specific tasks. For instance, one could attach the prompt "Is the following movie review positive or negative?" before the input sentence for sentiment analysis. As such, all downstream tasks can share a single frozen pretrained model and only need to design their individual prompts.

The most straightforward approach to craft a prompt is manual design, which we refer to as manual prompt. Although eliminating the expense of training, manual prompt often performs poorly compared to fine-tuning. To obtain strong task performance and efficiently serve multiple tasks, prompt-tuning technique [34], [54] emerges. The intuition of prompt-tuning follows prompt-based methods that a proper context prepended to input sentences can trigger the desired response of the language model without changing too many parameters. Instead of instantiating the prepended context with discrete tokens, prompt-tuning uses the trainable prompts as a replacement, also known as soft prompts

Fig. 2: Comparison of manual prompt, P-tuning, and P-tuning v2 in the language model pipeline. Manual prompt adds static prompt words to the input. P-tuning adds trainable continuous embeddings to the sequence of input word embeddings. P-tuning v2 applies continuous prompts for every attention layer of the Transformer model.

P-tuning [29], [39][2] and P-tuning v2 [38] are the state-of-the-art prompt-tuning techniques. Concretely, P-tuning applies non-invasive modification to the input. It replaces the input embeddings of the language models with differential and trainable embeddings. P-tuning v2 adds multiple extra parameters to the front of each attention layer. These parameters act on the output along with the other parameters of this layer. P-tuning has 0.01% trainable parameters per task compared to fine-tuning while P-tuning v2 has 0.1% to 3% trainable parameters [38]. Figure 2 illustrates the differences between the above three prompt techniques.

B. Task-agnostic Backdoors

Backdoor Attacks Against NLP. Backdoor attack, first proposed in [24], aims to force the model to predict inputs with triggers into a target class. In the context of NLP, a backdoored model classifies the clean text into the correct category while misclassifying the text containing a fixed n-tokens sequence $t = \{t_i\}_{i=1}^{m}$ (i.e., the trigger) to the attacker-specified label or the target label. We denote a normal Transformer model as $f(\cdot)$ parameterized by θ, and a clean dataset $\mathcal{D} = \{X, Y\}$, where $x = \{x_i\}_{i=1}^{n} \in X$, $y \in Y$ is the corresponding label. A backdoor attacker aims to get a model $f(\cdot)$, which classifies x to correct label y while misclassifies $x' = \{x_i\}_{i=1}^{n} \oplus \{t_i\}_{i=1}^{m}$ to target label y', where $\oplus$ denotes trigger injection operation.

Task-agnostic Backdoors. This type of attack injects backdoors in the pretrained models. The attacker in this scenario is agnostic to the downstream task, i.e., the attacker has no knowledge of downstream task datasets or model structures. With the growing popularity of model hubs such as HuggingFace[3], TensorFlow Model Garden[4], and ModelZoo[5], pretrained model backdoor becomes a practical security concern in real-world NLP systems. HuggingFace Hub, where anyone can upload or download models, now contains over 60K models. Reviewing each model's security is impractical for the model hub maintainer.

The state-of-the-art task-agnostic backdoors [84], [61], [76] rely on an output representation manipulation mechanism. Specifically, the attacker first pre-defines a vector and forces the outputs of the pretrained model to be as close to this vector as possible when the inputs contain triggers. We call this Pre-defined Vector PV in the following part of this paper. Formally, the backdoored pretrained model represents a clean input x normally, i.e., $f(x; \theta) \approx f(x; \theta')$. When the attacker injects a trigger t to the clean input x, getting x', the new representation turns out to be a PV, $f(x'; \theta') = v_t$, where v_t is the PV corresponding to trigger t. If the training of the downstream task does not remove the backdoor, then we will have $f(x'; \theta_{turn}) \approx v_t$. Therefore, the model prediction will be controlled by the trigger rather than the clean input.

Threat of Task-agnostic Backdoors. Prompt-tuning requires additional consideration for task-agnostic backdoors compared to fine-tuning, for two main reasons: (1) The pretrained model in prompt-tuning is typically used for multiple tasks, meaning that a single backdoored model can potentially compromise the security of multiple systems; (2) The property of prompt-tuning that freezes the pretrained model parameters makes the backdoor immune to catastrophic forgetting, where an artificial neural network will gradually forget previously learned information upon learning new information [46], [56]. To illustrate this point, we compare the effect of the training set size on the attack success rate in both fine-tuning and prompt-tuning scenarios (refer to Appendix A for more details). The experimental results show that the backdoors on the fine-tuning model gradually fade as the training set size increases, while the backdoors in prompt-tuning still persist.

C. Existing NLP Backdoor Defenses

Current NLP backdoor defenses generally consist of two steps: backdoor detection and backdoor removal.

Backdoor Detection. This step aims to determine whether a suspicious model is backdoored or not. The most widely used approach is trigger inversion. The general idea of trigger inversion is to backpropagate the gradient to the input and find the minimum amount of perturbation to change the predicted labels of any inputs to one target label. If one can invert a trigger from the suspicious model, the model is considered backdoored; otherwise, the model is benign. Trigger inversion is a mature technique in the computer vision domain [73], [67], [53]. However, in the NLP domain, one cannot directly backpropagate the gradient to the input to invert the trigger because of the inherent discontinuity of the sentences. Inverting word embeddings is also impractical. Due to the sparsity of the embedding space in NLP models, direct optimization of word embeddings usually results in either a failure to converge

[2] Lester et al. and Liu et al. proposed the idea of soft prompts almost simultaneously, and we use P-tuning in this paper to refer to these two works.

[3] https://huggingface.co/models

[4] https://github.com/tensorflow/models

[5] https://modelzoo.co

or the generation of invalid tokens. Recently, several studies began to tackle the challenge of trigger inversion in the text domain.

T-miner [3] proposes to train a sequence-to-sequence generative model for a target NLP model such that the generator model can perform a minimum transformation to any input to induce misclassification of the target model. PICCOLO [44] first transforms a target model to its equivalent and differentiable form and then optimizes a distribution vector denoting the likelihood of words being a trigger word. PICCOLO leverages the word discriminative analysis to check if the model is particularly discriminative for the inverted words. DBS [60] uses a similar method with PICCOLO that inverts the word probabilities distribution of the trigger. The difference is that DBS leverages a dynamically reducing temperature coefficient in the softmax function instead of word discriminative analysis to filter out local optima.

PICCOLO and DBS have demonstrated T-miner's ineffectiveness on large Transformer models, with a detection accuracy of less than 0.6. Furthermore, in our empirical analysis, we observe that PICCOLO and DBS struggle to converge on task-agnostic backdoors. To demonstrate the reasons, we use the method in [31] to visualize loss surfaces of PICCOLO on two types of backdoor attacks. Specifically, we choose BadNet [24] for the task-dependent backdoor and POR [61] for the task-agnostic backdoor. The visualization results are shown in Figure 11a and Figure 11b, respectively. PICCOLO has a much sharper transition from the center point to the perimeter on POR than on BadNet, which means PICCOLO has difficulty converging on POR. In our experiments, PICCOLO barely converges on task-agnostic backdoors (Section V-F). Since DBS shares a similar inversion method with PICCOLO, it faces the same convergence issues as PICCOLO on task-agnostic backdoors.

Backdoor Removal. Because of catastrophic forgetting, a backdoored model will gradually forget the previously learned backdoor behavior after being fine-tuned with a large amount of clean data; therefore, fine-tuning is a natural approach to removing backdoors. In [61], Shen et al. reveal that the trigger effectiveness drops significantly when the training dataset size exceeds 32K. Fine-prune [37] repair backdoored models by removing neurons that are not activated on the benign samples. NAD [35] proposes to utilize a teacher model to guide the fine-tuning process of the backdoored student model on clean data and make the attention of the student model align with that of the teacher model. However, all the above backdoor removal approaches require auxiliary model updating and storage, which inevitably hamper the modularity and low storage nature of prompt-tuning. We aim to propose a defense scheme that conforms to the prompt-tuning characteristics, such as low storage and high availability.

A different approach to backdoor removal focuses on the input side, with the goal of eliminating triggers from the inputs. Techniques such as ONION [52] remove triggers, while methods like STRIP [22] and RAP [77] reject any inputs containing triggers. ONION assumes the triggers increase the text perplexity; however, task-agnostic backdoors can work with arbitrary trigger designs, such that studying a trigger-agnostic defense method is more important. In contrast to STRIP and RAP, which reject and discard inputs with triggers, LMSanitator can effectively remove the triggers and enable the model to classify poisoned inputs into the correct class.

III. Attack Methodology

Previous task-agnostic backdoor attacks against pretrained models mainly target fine-tuning scenarios. In this section, we adapt the task-agnostic backdoor attacks to the prompt-tuning scenarios.

A. Threat Model

Attackers' Goal. We consider an attacker aiming to inject backdoors into a pretrained model such that its downstream prompt-tuning model behaves at the attacker's will on a triggered input. The backdoored model should maintain utility to be stealthy. In particular, the downstream model built based on the backdoored language model should be as accurate as a downstream model built based on a clean language model.

Attackers' Knowledge. We assume the attacker can query the downstream task system but is completely agnostic to the downstream tasks, which means the attacker has no access to the training dataset and no knowledge of prompt-tuning model architecture (including the head and prompt network). One example scenario is that an adversary publishes a backdoored model to the HuggingFace model hub and claims it is optimized for spam detection tasks. A spam detection service provider may download and use this model in their detection system. The adversary queries the system to determine whether it is backdoored. If yes, the adversary then selects an appropriate trigger and inserts it into their email to bypass the spam detection system. It is worth noting that the adversary can fine-tune an open-source pretrained model and re-release it, circumventing the necessity of training from scratch. This substantially reduces the expenditure associated with task-agnostic backdoors. In our empirical study, we are able to poison a RoBERTa-base model in less than 25 minutes and a RoBERTa-large model in under 70 minutes using an RTX 3090 GPU.

B. Attack Details

We follow the attack approaches in [84], [61], [76]. The general idea is to embed backdoors by mapping inputs with the trigger to a certain representation vector. Instead of binding a trigger to a specific target label in traditional backdoor attacks, a task-agnostic backdoor aims to associate the trigger with a certain output representation (we call it PV in this paper). For example, an adversary can predefine an output representation for the [CLS] token to attack text classification tasks or predefine an output representation for all normal tokens to attack NER tasks. The specific representation is then mapped by the head to a specific label.

The attacker has two goals—effectiveness goal and utility goal. The effectiveness goal means that the attacker wants to make some certain tokens' feature output of the pretrained model as close to PV as possible when the input contains a trigger. Also, to prevent the backdoor from being detected by the user, the attacker wants the model to produce normal output representations when the input is clean. Task-agnostic attacks translate these two goals into two optimization problems. For the effectiveness goal, task-agnostic attacks define an

effectiveness loss L_e that calculates the distance between target output representations of the model and the PV. NeuBA [84] and POR [61] use MSE loss while BToP [76] uses pairwise distance. For the utility goal, task-agnostic attacks define a utility loss L_u that keeps the model function properly for clean inputs. NeuBA and BToP make the model do all-mask task at the time of inputting clean sentences. POR adds a reference model to form a pseudo-siamese network and restricts the distance between the target model output and reference model output when the input is clean. Formally, task-agnostic backdoor attacks follow the optimization below:

$$\arg\min \lambda_e \mathop{\mathbb{E}}_{x \in D} L_e(f(x; \theta); PV) + \lambda_u \mathop{\mathbb{E}}_{x \in D} L_u(f(x; \theta)), \quad (1)$$

where λ_e and λ_u are two hyperparameters to balance these two loss terms. The attacker usually injects multiple orthogonal PVs into the model to ensure at least one of them can be mapped to the target label.

C. Disscusion

Note that there are other existing studies on backdoors against prompt-tunning; however, most of them focus on task-specific backdoors. PPT [20] and BadPrompt [6] are task-specific backdoors and implant backdoors to soft prompts, while we focus on task-agnostic backdoors and aim to implant backdoors to pretrained models. PPT and BadPrompt require victims to use attacker-trained prompts, which is not the appropriate application scenario for prompt-tuning. The primary goal of prompt-tuning is to facilitate users with limited resources to use large models, and users can easily train prompts locally instead of downloading them from the Internet. Furthermore, both PPT and BadPrompt require the attacker to have access to the downstream dataset or a domain shift dataset, which is unpractical in privacy scenarios.

Xu et al. [76] also investigate the impact of task-agnostic attacks on the prompt-based learning paradigm. However, they focus on prompt-based fine-tuning (PFT), which utilizes manual prompts and optimizes the entire model; in contrast, our research studies prompt-tuning, a distinct approach that integrates trainable soft prompts into the pretrained model and optimizes only soft prompts. We aim to explore the attack effects and defense methods of task-agnostic backdoors when the pretrained model is frozen.

IV. Defense Methodology

A. Method Overview

Design Intuition. Trigger inversion is a difficult problem in the text domain because of its inherent discontinuity. Although PICCOLO replaces the tokenizer and embedding layer with an equivalent word encoding, it is still limited by word vocabulary size. Our key idea is that since it is difficult to invert the input, it might be easier to invert the output. Inverting output can circumvent the problem of the infeasibility of input layers in NLP models, and avoid increasing the number of model layers. Task-agnostic backdoors map one trigger to a PV, which acts as an outlier in the feature space. We find inverting PVs converges more easily on task-agnostic backdoors than inverting triggers. If a legitimate PV can be inverted from a pretrained model, we consider the model contains task-agnostic backdoors. Then, we can determine whether the input contains a trigger by monitoring the similarity between the pretrained model's output and the found PV.

Pipeline. LMSanitator consists of three steps: PV mining, PV filtering, and PV monitoring, as shown in Figure 3. PV mining uses an iterative approach to mine PVs implanted in the pretrained model. PV filtering filters illegal PVs found in the first step. If the task-specific model developer wants LMSanitator to do backdoor detection, the developer only needs to perform the first two steps and only needs to run PV mining for a small number of iterations. If the detection result is backdoored, but the developer still wants to use the backdoored pretrained model to build a trusted prompt-tuning model, the developer needs to run PV mining for more iterations to get a PV set. We call this operation PV searching. Then the developer needs to proceed to the third step (PV monitoring). The third step detects and removes triggers based on the PV set during inference. All of the above steps only require the defender to possess a small clean sentence dataset (containing 2000 sentences in our experiments), which is easy to obtain from the Internet. Next, we present the design of each step in detail.

B. Step I: PV Mining

PV mining aims to invert the attacker-designed PVs. We have discussed (see Section II-C) that letting the model misclassify inputs to a specific class does not work. Thus, the difficulty lies in defining exception output (i.e., PV) and designing practical optimization functions. To solve this dilemma, we propose two losses (distance loss and diversity loss) based on our two key observations. In this subsection, we first introduce these two observations (more observation support experiments can be found in Appendix C) and then present our loss function design. Last, we propose two novel mechanisms to improve inversion efficiency.

Observations. The first observation is that the sentence with the trigger will act as an outlier in the feature space of the backdoored Transformer model. We train a backdoored BERT-base-cased model using a POR attack and randomly select a sentence from SST-2 [63] dataset as the clean input. Then, we insert the trigger words and 200 other random non-trigger words into the clean sentence and record [CLS] outputs of the backdoored model. The visualization results are shown in Figure 4a. We formally define observation I as:

Observation I: Let $f(\cdot;\cdot)$ denote a backdoored language model, x denotes a clean sentence, and t denotes a trigger from trigger set T. We use $dis(x_i; x_j)$ represent the distance between $f(x_i; \theta)$ and $f(x_j; \theta)$. Then for arbitrary x, we have

$$dis(x \oplus w_i; x \oplus t) \approx dis(x \oplus w_i; x \oplus w_j), \quad t \in T, w_{i,j} \notin T. \quad (2)$$

The second observation is that the feature distance between two clean sentences on a backdoored model will shrink if the same trigger is inserted. We randomly select 200 sentences from SST-2 dataset and insert trigger words to get another 200 sentences. We input these 400 sentences to the backdoored model and record [CLS] outputs. Experimental results after

Fig. 3: LMSanitator pipeline. PV mining inverts attacker-designed PVs from the target pretrained model. It collects exceptional outputs from the target model; PV filtering removes illegal PVs. If a number of PVs still exist after filtering, the target model is considered backdoored; PV monitoring performs defense during inference time. It detects and removes triggers in the input

(a) (b)

Fig. 4: Visualization of models' output with/without triggering the backdoor. (a) The sentence in which the trigger word is inserted is far from other words in the feature space. (b) Different sentences with the same trigger focus on one point.

PCA dimensionality reduction are shown in Figure 4b. We can see that all sentences with triggers concentrate on one point, although they were previously scattered. We formulate observation II as follows:

Observation II: In a backdoored language model $f(\cdot;\ )$, let x_i and x_j denote two clean sentence, we have

$$dis\ (x_i\ \ t; x_j\ \ t)\ \ dis\ (x_i; x_j);\quad t \in T. \tag{3}$$

Inversion Losses. Through Observation I, we know that trigger words can significantly change the output of pretrained models, but other words do not. Inspired by this, we build a pseudo-siamese neural network to find trigger embeddings. Specifically, we first make a copy of the target model as the auxiliary model. Then we freeze the parameters of the target and auxiliary models and add a trainable soft prompt to the target model like P-tuning. Concretely, once the input passes through the embedding layer, resulting in a token embedding sequence, we split the sequence at a randomly chosen position and insert trainable embeddings at this split point. After reassembling the token embeddings, the refined sequence is forwarded to the subsequent layers. The input texts will be fed to both the target and the auxiliary model. Our first optimization objective—distance loss—aims to increase the distance between the target model output and the auxiliary model output. We use MSELoss to characterize the distance between these two vectors, formally defined as:

$$L_D = \ \mathop{E}_{x\ D} MSE(F_{tar}; F_{aux}); \tag{4}$$

where F_{tar} and F_{aux} are feature vectors generated by the target model and auxiliary model, respectively.

Based on Observation II, we give our second optimization objective—diversity loss. This loss term aims to make feature outputs within a batch as similar as possible, i.e., to reduce the output diversity. We use Shannon entropy to define the diversity loss as:

$$L_{div} = \ Entropy\ (Stack(F^{x}_{tar}; x\ \ Bg))^T; \tag{5}$$

where "Stack" means concatenating vectors in a new dimension. Note that we transpose the feature matrix before computing Shannon entropy. This is because we are not trying to reduce the diversity of each feature vector but the diversity of each dimension of feature vectors within a batch. Since higher Shannon entropy means lower diversity, a negative sign needs to be added when calculating diversity loss.

After defining the above two loss terms L_D and L_{div}, we formulate PV inversion as an optimization problem and update the soft prompt with the following optimization target:

$$\arg\min_{p} L = \ _D\ L_D + \ _{div}\ L_{div}; \tag{6}$$

where $_D$ and $_{div}$ are two parameters to balance the two loss terms. When L drops below a certain threshold T_L, we consider that a backdoor is found. At this point, we record the target model's output and the soft prompt's parameters for the next step.

Although we have designed the inversion loss complying with task-agnostic backdoor features, inverting backdoors beneath a pretrained model is still a tricky task. That is, PV mining's loss surface is flat in most cases. In the RoBERTa-large experiments, if we inject one PV in the model, trigger inverting only converges once in 20 training sessions on average. To solve this problem, we propose two novel mechanisms: *fuzz training* and *adaptive learning rate*.

Fuzz Training. This design is inspired by fuzz testing, a technique widely used in the software security domain. Its main idea is to input automatically or semi-automatically generated random data into a program, monitor the program for exceptions such as crashes, and assert failures to find possible errors such as memory leaks. Some fuzz techniques use optimizations to help the fuzzer trace more execution paths and find more bugs [23], [9], [13]. Similar to fuzz testing, we

use a test dataset and different random seeds to enhance inversed training. The test dataset contains clean sentences. Random seeds affect the initialization of soft prompt parameters. We initialize the soft prompt using embeddings in the vocab indexed from 70 seed to 70 seed + l_{sp} 1, where l_{sp} is the length of the soft prompt. To find as many PVs as possible instead of converging to the same PV all the time, we add an extra loss term—path loss

$$L_P = \max_i(\mathrm{MSE}(F_{tar}; c_i)); \quad (7)$$

where c denotes a found candidate PV vector. We first calculate the MSE value between F_{tar} and each discovered PV candidate and the maximum of them. We do not track gradients for this process. Afterward, we recalculate this maximum MSE with gradient tracking. Finally, we take the negative of this value as our path loss. Path loss increases as the output of the target model moves closer to Path loss is a dynamic loss, and we do not know which candidate to compute it in advance. If there is no PV candidate, the path loss is set to 0. In summary, our training objective becomes to:

$$\arg\min_p L = {}_D L_D + {}_{div} L_{div} + {}_P L_P; \quad (8)$$

where $_P$ is a parameter to adjust the weight of the path loss.

Adaptive Learning Rate. As mentioned before, PV mining's loss space is at in most cases. Therefore, we need to set a large learning rate; otherwise, the backpropagated gradients will be too small to update soft prompts effectively. However, a large learning rate prevents the model from converging to the optimum. To solve this problem, we adjust the learning rate according to gradients. At first, we set a large learning rate lr_0. We detect gradients of soft prompt parameters after each iteration. When the gradient of one parameter is larger than a threshold T_{grad}, we reset lr to $0{:}01 lr_0$. We summarize the process of PV mining in Algorithm 1.

Algorithm 1: PV Mining

Input: $_D$, $_{div}$, $_P$, T_L, T_g, lr_0, L_{max} (max number of fuzz loops)
Output: C (PV candidate list), P (soft prompt list)

```
C ← {}, P ← {}
seed ← 0
p ← init(seed)                          // initialize soft prompt
for l = 0 ! L_max − 1 do                // fuzz loop
    lr ← lr_0
    for epoch = 0 ! 4 do
        F_tar ← computeFeature(p)
        L ← computeLoss(F_tar, _D, _div, _P)
        @p ← r_p L
        p ← p − lr · @p
        if max(@p) > T_g then
            lr ← 0:01 · lr_0
        end
    end
    if L < T_L then
        C.append(F_tar)
        P.append(p)
    end
    seed ← seed + 1                     // mutate seed
    p ← init(seed)
end
```

C. Step II: PV Filtering

The exceptional outputs obtained by PV mining may not be caused by the backdoor but by out-of-range (too large or too small) soft prompts. Therefore, given a set of PV candidates and their corresponding soft prompts, we need first to justify whether the value of a dimension in soft prompts is out of the range of values of the embedding layer parameters. If this happens, we remove the corresponding PV candidates.

Second, we find a situation where L_D decreases, while L_{div} remains high. This phenomenon also occurs in clean models. In this case, soft prompt increases the distance between F_{tar} and F_{div}, but does not reduce the diversity of F_{tar}, i.e., the distance between two F_{tar} in a batch is high, which is contrary to observation II. Although the reduction of L_D makes the total loss lower than T_L, this situation does not belong to the task-agnostic backdoor. To filter out this type of illegal PVs, we design an additional threshold T_{div} and remove PV candidates whose L_{div} are higher than T_{div}.

We get the final PV set after the two-step filtering. We repeat the fuzz training L_{max} times to make a decision. If the PV set is still empty, we consider the target model is clean. Otherwise, the target model contains task-agnostic backdoors.

D. Step III: PV Monitoring

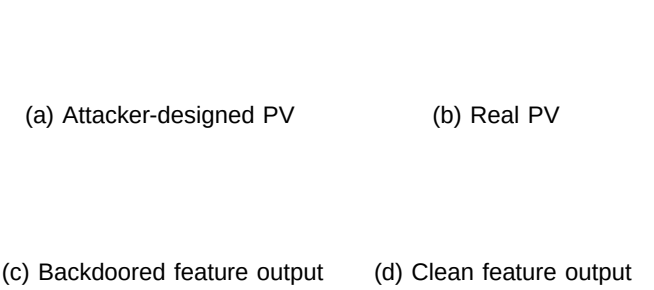
After obtaining the PV set, a simple trigger detection method is first to let inputs go through the pretrained model before feeding them into the task-specific model and then calculate the similarity between pretrained model feature outputs and PVs. This is effective but time-consuming. Because an input needs to go through the Transformer model twice. We find that in the prompt-tuning model, placing the monitor on the output side is also effective. Immuning to catastrophic forgetting, language models in prompt-tuning models output vectors very close to PVs when the input contains a trigger.

We train a backdoored RoBERTa-base model using the POR attack. The PV we designed is shown in Figure 5a. In fact, model outputs cannot be exactly the same as attacker-designed PV, and the real PV is shown as Figure 5b. We find that the feature output of the backdoored model for triggered input is consistent with PV in terms of positive and negative signs even after prompt-tuning. In contrast, the sign distribution of clean feature outputs is random.

Trigger Detection. We propose a more efficient sign-based trigger detection method. The defender first converts PVs in the PV set into sign tuples (e.g., $[0{:}5; 0{:}1; \ ; 0{:}7] \to [+; +; \ ;]$) to get a PV sign set and puts a monitor at the output side of the Transformer model. For each feature vector $F_{LM} \in \mathbb{R}^d$ output by the language model in the reference process, the monitor will count the number of F_{LM}'s signs that match tuples in the PV sign set. Let F^n_{LM} and PV^n represent the n-th dimension value of the feature vector and PV, respectively. The match number N_{match} can be expressed

TABLE I: Datasets taxonomy and statistics.

Granularity	Task Type	Dataset	Balance	#Classes	#Inputs	Train	Valid	Test
Sentence Classification	Natural Language Inference	RTE [72]	even	2	2	2,490	277	-
	Question Answering	BoolQ [14]	even	2	2	9,427	3,270	-
	Topic Classification	AG News [82]	even	4	1	6,000	2,000	7,600
	Sentiment Analysis	Yelp-5 [82]	even	5	1	6,000	2,000	2,000
	Spam Detection	Enron spam [48]	even	2	1	6,000	2,000	2,000
		SMS spam [16]	uneven	2	1	4,458	558	558
Token Classification	Named Entity Recognition	CoNLL04 [7]	uneven	9	1	8,936	2,012	1,671
		OntoNotes 5.0 [51]	uneven	37	1	37,946	5,037	5,053



(a) Attacker-designed PV (b) Real PV

(c) Backdoored feature output (d) Clean feature output

Fig. 5: The positive and negative signs of the backdoored output feature are consistent with the PV.

as the following equation:

$$N_{match} = \sum_{n=1}^{d} 1\{sign(F_{LM}^{n}) = sign(PV^{n})\}; \quad (9)$$

where $1\{\cdot\}$ denotes the indicator function that is 1 when $\cdot$ is true and 0 when $\cdot$ is false. If N_{match} exceeds a specific value T_{match} (we recommend 0.8d as a rule of thumb), the monitor considers that the input contains a trigger.

Trigger Removal. Once the input text is determined as triggered, we want to determine further which words are the triggers. This allows us to remove triggers and let the model classify the input to the correct class. We use a sliding window to mark the candidate trigger. The starting length of the sliding window is set to 1. The sliding window slides from the beginning to the end of the input sentence with a stride of 1. If the input has two sentences, the sliding window must slide over each. When the sliding window slides to a position, we consider words inside it as candidate triggers. After that, we remove all candidate triggers in the input sentences. If the input has two sentences, we need to remove candidate triggers in the other sentence. After removing candidate triggers, we use trigger detection to detect whether the input is triggered. If the answer is 'no', the model outputs the classification results; otherwise, the sliding window moves to the next position. We gradually increase the sliding window length for traversal until trigger detection returns false or the sliding window length reaches the maximum. Due to the space limitation, we place the details of the trigger removal algorithm in Appendix D of our technical report [74].

Our PV monitoring method only needs to add one step of trigger detection when the input is clean, which minimizes the computing consumption of defense. We find that our PV monitoring method rarely identifies clean inputs as trojaned, which means that our method has no impact on model accuracy. The theoretical analysis of LMSanitator's effect on clean inputs can be found in Appendix E of our technical report [74]. When the input is trojaned, PV monitoring has a complexity of $O(l_t \cdot l_i)$, where l_t and l_i is the length of the trigger and input respectively.

V. Evaluation

In this section, we first evaluate LMSanitator's end-to-end performance on sentence classification tasks. Second, we explore the backdoor detection capability of LMSanitator. Third, we conduct PV searching experiments to show that LMSanitator can find most of PVs. Fourth, we conduct an ablation study to illustrate the necessity of LMSanitator's mechanisms. Fifth, we compare LMSanitator with baselines.

Additionally, we investigate the following issues; however, due to space limitation, the associated results are placed in our technical report [74]: (1) We investigate attack performance and LMSanitator's end-to-end performance on two NER datasets (Appendix J of [74]). (2) We empirically measure the time required by LMSanitator, including backdoor detection time, PV searching time, and trigger detection time (Appendix K of [74]). (3) We analyze hyperparameters' effect on the performance of LMSanitator. (Appendix L of [74]).

A. Experimental Setup

Prompt-tuning Datasets. To demonstrate the generality of our approach, we perform experiments on various types of downstream tasks, including 6 sentence level classification tasks and 2 token level classification tasks. In addition to single-sentence classification tasks, our datasets also contain two sentence-pair classification tasks (RTE and BoolQ), whose input consists of two sentences that are spliced by [SEP]. Table I summarizes the taxonomy and statistics of the used datasets. We refer the readers to Appendix F of our technical report [74] for details of these datasets.

Victim Models. We choose two popular types of language models, BERT [18] and RoBERTa [43] for end-to-end and backdoor detection evaluation. For each type, we choose two different sizes, large and base. The large models have 24 attention layers and a hidden size of 1024, while the base models have 12 attention layers and a hidden size of 768. For PV searching evaluation, we additionally choose four types of language models, DeBERTa [25], ALBERT [28], ERNIE [83], and XLNet [79].

Evaluation Metrics. For attacks, we report the clean model accuracy (ACC_{clean}), backdoored model accuracy ($ACC_{backdoor}$), and attack success rate (ASR). For defenses, we report the attack success rate before and after applying defenses. ACC_{clean} and $ACC_{backdoor}$ measure the classification accuracy of a clean downstream classifier and a backdoored downstream classifier with clean input, respectively. An $ACC_{backdoor}$ close to ACC_{clean} indicates the backdoored model does not affect the normal task. ASR measures the fraction of triggered inputs that are misclassified to a wrong class by a backdoored downstream classifier. We insert each attacker-chosen trigger in turn and consider the attack successful if one can cause the misclassification.

Since some datasets are unevenly distributed, we also report the F1 scores and the model accuracy. We also report the weighted ASR for sentence classification tasks and F1 drop for token-classification tasks along with ASR to measure attack effectiveness. The weighted ASR is the average of ASRs over each input class. F1 drop measures the value of F1 drops after inserting a trigger.

Attack Setup. We use BToP, NeuBA, and POR to generate the backdoored pretrained models. We inject six triggers for each pretrained model. The trigger set we use is {`cf', [`mn', `tq', `qt', `mn', `pt']. Note that although all of these triggers are single-word, some models' tokenizers recognize them as multiple tokens. For example, the BERT tokenizer decomposes `cf' into `c' and `##f', while ALBERT tokenizer decomposes `tq' into `_', `t' and `q'. We obtain six orthogonal PVs by the POR-2 method proposed in [61]. In particular, we divide the output vector into four equal parts. Then, we use different $(1, -1)$ combinations to fill them. Each trigger corresponds to one PV. For attack datasets, we follow the choices made in the original papers. Specifically, BToP and POR use WikiText [47], and NeuBA uses BookCorpus [85]. We sample 5000 plain sentences from the attack dataset for each trigger to compute effectiveness loss and another 5000 plain sentences to compute utility loss.

Since BToP targets [MASK] token while NeuBA and POR target [CLS] token, we use BToP to attack P-tuning models and use NeuBA and POR to attack P-tuning v2 models.

Defense Setup. LMSanitator only requires the defender to have a small clean dataset. We sample 2000 plain sentences from WikiText to form the defense dataset. Note that the defense dataset and attack dataset do not overlap. LMSanitator has the following parameters: $_D$, $_{div}$, $_P$, T_L, T_{div}, T_{grad}, T_{match}, and l_{sp}. Unless otherwise mentioned, we use the following default settings: $_D = 1$, $_{div} = 1$, $_P = 0.5$, $T_{div} = 3.4 \times 10^{-6}$, $T_{grad} = 5 \times 10^{-3}$, $T_{match} = 0.8d$, and $l_{sp} = 7$, where d is the hidden dimension of the target pretrained model. The value of T_{div} depends on the training batch size, which is 32 in our setting. If a user wants a larger batch size, T_{div} should be adjusted downwards. We empirically find that if PV mining cannot converge in the first two epochs of one fuzz loop, it will unlikely converge in the following two epochs. Therefore, to speed up PV mining, we check whether the model converged before the third epoch's start. If it converges, we continue the training; otherwise, we go to the next fuzz loop. In addition, we find that L_D decreases before L_{div}. The decrease of L_{div} occurs mainly in the last two epochs. Therefore, we use T_L to constrain only L_D, and let PV filtering to constrain L_{div}. We set $T_L = 0.1$ by default.

B. Results on Sentence Classification Tasks

In this section, we evaluate sentence classification tasks to illustrate the attack and defense effectiveness.

Setup. We train downstream prompt-tuning models using the clean pretrained models and the backdoored pretrained models, respectively, and test their accuracy on their test sets. The hyperparameters we used are illustrated in Appendix G (Table 11) of our technical report [74]. P-tuning needs some manual work to design initial prompts and verbalizers. The initial prompts and verbalizers we used are displayed in Appendix H of [74]. For SMS spam, whose dataset is unbalanced, we also compare macro-F1 scores. We insert one trigger into the test input to test the ASR. For tasks like RTE and BoolQ where the input has two sentences, we insert the trigger to the longer sentence for stealthiness. We vary the seed and calculate the average of three trials to get the final results.

Attack Stealthiness. Table II compares the clean model accuracy and the backdoored model accuracy on 6 sentence-level classification tasks. In general, we observe that existing task-agnostic backdoors can preserve the accuracy of downstream prompt-tuning classifiers. In particular, the differences between the backdoored and clean model accuracy are less than 1% in most cases. We further observe that the accuracy degradation of the P-tuning models is more significant than that of the P-tuning v2 models. This is because P-tuning v2 adds more parameters than P-tuning, thus relying less on the pretrained model parameters. We observe that NeuBA significantly impacts the accuracy of RTE and Yelp-5 tasks. This indicates that NeuBA's utility loss cannot fully preserve the model's functionality. We suspect that part of clean sentences can also trigger the backdoor in NeuBA-attacked models.

Attack Effectiveness. Data on the left of Table III shows the ASR of existing task-agnostic backdoors on prompt-tuning. In general, we observe that the task-agnostic backdoors achieve high attack success rates in most cases. Out of our total 72 sets of experiments, 39 sets achieve ASR of higher than 99%. Comparing P-tuning and P-tuning v2, we observe that P-tuning is more vulnerable to backdoor attacks than P-tuning v2. Concretely, the percentage of achieving 99% ASR on P-tuning is 0.83 (20/24), while the percentage of achieving a 99% ASR on P-tuning v2 is only 0.40 (19/48). In general, these results demonstrate the vulnerability of prompt-tuning to task-agnostic backdoors.

Defense Effectiveness. We first use PV mining and PV filtering to obtain the PV set. Since we want to test the best-case performance of the trigger, we add attacker-designed PVs that are not found in PV mining to the PV set when testing the effectiveness of PV monitoring. This is reasonable because the attacker does not know which PVs are found by the defender in practice. Also, the experiments in Section V-D show that our PV mining can find attacker-designed PVs fully on most language models. Data on the right of Table III shows the ASR after deploying our LMSanitator defense. The end-to-end experimental results demonstrate that our defense approach can effectively reduce ASRs of task-agnostic backdoors on prompt-tuning. Out of our total 72 experiments, ASR decreases

TABLE II: Model accuracy on sentence classification tasks. Numbers on the left/right refer to the clean/backdoored model accuracy.

Prompt	Attack	Victim Model	RTE ACC	BoolQ ACC	AG News ACC	Yelp-5 ACC	Enrron spam ACC	SMS spam ACC	SMS spam F1
P-tuning	BToP	RoBERTa-large	70.51 | 68.35	69.53 | 64.55	90.27 | 88.65	60.84 | 49.88	96.18 | 95.76	99.10 | 99.28	96.50 | 97.19
		RoBERTa-base	61.61 | 62.94	61.85 | 62.14	88.90 | 88.82	55.69 | 50.14	95.88 | 94.49	96.50 | 97.19	98.13 | 98.35
		BERT-large-cased	58.60 | 55.23	62.16 | 62.32	88.67 | 86.98	49.75 | 47.63	92.27 | 90.88	99.46 | 99.16	97.90 | 96.68
		BERT-base-cased	54.99 | 54.99	62.29 | 62.09	88.10 | 88.47	47.16 | 45.24	93.78 | 92.35	99.58 | 99.40	98.36 | 97.68
P-tuning v2	NeuBA	RoBERTa-large	87.00 | 85.92	83.70 | 83.49	95.00 | 95.00	67.14 | 63.57	99.29 | 98.29	99.82 | 99.73	99.31 | 98.95
		RoBERTa-base	76.90 | 70.15	78.69 | 78.81	92.57 | 93.14	62.86 | 60.43	98.71 | 98.43	99.64 | 99.55	98.61 | 98.28
		BERT-large-cased	75.45 | 71.12	73.12 | 73.06	92.57 | 92.71	58.00 | 56.71	98.86 | 98.71	99.55 | 99.55	98.25 | 98.26
		BERT-base-cased	71.84 | 69.79	72.02 | 71.90	92.00 | 91.71	57.71 | 55.71	98.57 | 98.71	98.83 | 99.28	95.53 | 97.18
	POR	RoBERTa-large	87.00 | 86.28	83.70 | 83.43	95.00 | 94.57	67.14 | 63.57	99.29 | 99.29	99.82 | 99.64	99.31 | 98.60
		RoBERTa-base	76.90 | 75.45	78.69 | 78.20	92.57 | 91.86	62.86 | 60.00	98.71 | 98.57	99.64 | 99.37	98.61 | 97.54
		BERT-large-cased	75.45 | 75.81	73.12 | 73.09	92.57 | 92.14	58.00 | 57.71	98.86 | 98.86	99.55 | 99.55	98.25 | 98.25
		BERT-base-cased	71.84 | 70.40	72.02 | 72.78	92.00 | 91.57	57.71 | 57.00	98.57 | 98.43	98.83 | 98.83	95.53 | 95.47

TABLE III: Attack success rate on sentence classification tasks. Numbers on the left/right refer to ASR without/with defense.

Prompt	Attack	Victim Model	RTE ASR	BoolQ ASR	AG News ASR	Yelp-5 ASR	Enron spam ASR	SMS spam ASR	SMS spam WeightedASR
P-tuning	BToP	RoBERTa-large	100.0 | 0.00	99.92 | 0.05	100.0 | 0.00	99.59 | 0.00	76.33 | 26.69	99.85 | 0.18	99.91 | 0.63
		RoBERTa-base	100.0 | 0.57	99.88 | 0.00	100.0 | 0.04	99.36 | 0.22	90.03 | 0.74	99.88 | 0.00	99.93 | 0.00
		BERT-large-cased	100.0 | 0.00	99.95 | 0.10	99.70 | 0.27	99.35 | 0.00	87.51 | 1.28	99.94 | 0.09	99.97 | 0.38
		BERT-base-cased	100.0 | 0.00	99.87 | 0.05	100.0 | 0.00	99.34 | 0.00	91.81 | 0.00	100.0 | 0.00	100.0 | 0.00
P-tuning v2	NeuBA	RoBERTa-large	98.25 | 2.18	25.50 | 5.19	99.91 | 2.53	16.55 | 13.67	13.10 | 2.29	67.09 | 0.18	80.53 | 0.40
		RoBERTa-base	100.0 | 8.34	99.53 | 10.29	99.91 | 5.42	97.37 | 14.28	47.44 | 6.05	94.95 | 0.63	97.11 | 1.02
		BERT-large-cased	11.11 | 9.72	19.72 | 8.26	99.96 | 6.80	82.63 | 17.69	59.78 | 5.59	100.0 | 7.21	100.0 | 9.24
		BERT-base-cased	100.0 | 10.22	97.99 | 7.77	99.93 | 23.91	81.00 | 37.13	15.62 | 3.72	99.73 | 0.72	99.85 | 1.97
	POR	RoBERTa-large	100.0 | 0.00	92.88 | 0.37	99.98 | 0.18	86.66 | 23.61	26.63 | 1.21	98.83 | 0.09	99.33 | 0.35
		RoBERTa-base	99.01 | 8.42	100.0 | 3.63	98.10 | 2.92	100.0 | 5.35	98.78 | 2.08	64.98 | 0.45	78.13 | 1.03
		BERT-large-cased	99.36 | 0.70	100.0 | 0.36	74.00 | 0.00	18.76 | 0.95	86.94 | 1.87	21.62 | 0.27	52.40 | 0.78
		BERT-base-cased	100.0 | 0.00	100.0 | 0.06	99.97 | 0.00	68.09 | 0.12	80.34 | 6.21	47.01 | 0.00	49.67 | 0.00

to less than 5% in 50 sets of experiments and less than 1% in 40 sets of experiments. Only 5 groups of experiments maintain ASR after using our defense approach. LMSanitator performs better on P-tuning because P-tuning adds fewer parameters than P-tuning v2. Among the three backdoor attacks, our approach encounters performance degradation on NeuBA. We speculate that it may be because some clean sentences are also mapped to a PV after the NeuBA attack, such that an unknown PV is output after injection of one trigger, causing our trigger detection algorithm to fail.

Visualization and Analysis. The success of task-agnostic backdoors and our defense is not always guaranteed. To better understand these phenomena, we visualize the distribution of PV match rates. Due to the space limitation, we refer the readers to Appendix I of [74] for the visualization results. In most cases, poisoned inputs exhibit a high match rate (> 0.9), while clean inputs typically have a match rate below 0.7. Therefore, setting T_{match} to 0.8 can yield FRR (False Rejection Rate) and FAR(False Acceptance Rate) values close to 0 in these cases. When the match rates of poisoned and clean inputs are mixed up, the ASR of the backdoor is very low, and defense is unnecessary. In cases where the poisoned match rates are scattered, we can significantly reduce ASR by filtering points with higher match rates.

C. Effectiveness of Backdoor Detection

Previous experiments demonstrate the end-to-end defense effectiveness of LMSanitator. In this section, we explore the backdoor detection capability of LMSanitator.

Setup. We build clean models by fine-tuning the original pretrained models downloaded from HuggingFace. We build 120 clean models for each architecture by varying the dataset and seed. We use the 6-sentence classification datasets with 20 seeds each. We randomly select 30 of these 120 clean models to measure the detection accuracy of LMSanitator for each attack method. We use NLTK [45] to generate 200 random triggers. Half of them are single-word, and the other half are two-word. For each backdoored model, we randomly select 6 of these 200 triggers as attack triggers. Then we use 4 attack approaches to build 30 backdoored models for each model architecture. We filter out the models whose losses do not fully converge until we get 30 successfully attacked models. We perform 30 fuzz loops (i.e., L_{max} = 30) on each test model and consider the model as backdoored if a legitimate PV can be found. Empirically, we find that models with different architectures have different sensitivities to T_{div}, and learning rate. Thus, we first train 5 shadow models for each architecture using BToP and use them to adjust these hyperparameters. These hyperparameters we use in backdoor detection are shown in Appendix G (Table 9) of our technical report [74].

Results. We use false positives (clean models flagged as backdoored), false negatives (backdoored models flagged as clean), and accuracy (fraction of correctly flagged models) as our evaluation metrics. The detection results are summarized in Table IV. Across all 4 attacks, we achieve an average of 92.8%

TABLE IV: Backdoor detection performance of LMSanitator. FP = false positive, FN = false negative.

Victim Model	Defense: Changing the Target Token										Average Acc
	[CLS]				[MASK]			Normal Token			
	FP	FN (POR)	FN (NeuBA)	Acc	FP	FN (BToP)	Acc	FP	FN (POR-NER)	Acc	
RoBERTa-large	7/60	2/30	11/30	83.3%	1/30	1/30	96.7%	0/30	13/30	78.3%	85.4%
RoBERTa-base	9/60	0/30	0/30	92.5%	3/30	0/30	95.0%	1/30	0/30	93.3%	94.6%
BERT-large-cased	8/60	0/30	5/30	89.2%	0/30	0/30	100.0%	0/30	3/30	95.0%	93.3%
BERT-base-cased	0/60	0/30	2/30	98.3%	0/30	0/30	100.0%	0/30	3/30	95.0%	98.0%

Fig. 6: PV searching results against different attacks. Attack PVs are the true attacker-designed PVs. Unintended PVs are PV found by LMSanitator but not prede ned by the attacker. The dotted lines indicate the position where the number of PVs is 6.

Fig. 7: Study of an unintended PV. Each row is a pretrained model feature output. We can see that the unintended PV is a superposition of two attack PVs.

detection accuracy. In general, LMSanitator can accurately ag backdoored models. We nd that LMSanitator is more likely to generate FN on large models and FP on base models. This indicates that larger models are harder to converge. We recommend users increase fuzz loops on large models and decrease fuzz loops on small models when using LMSanitator for backdoor detection.

D. Effectiveness of PV Searching

After determining that a pretrained model contains task-agnostic backdoors, the user would want to nd as many PVs as possible to achieve better defense effectiveness in later PV monitoring. Section V-C proves that LMSanitator has a high backdoor detection accuracy. In this section, we demonstrate that LMSanitator can also nd most of attacker-designed PVs.

Setup. We use the attack setup described in Section V-A to train 42 backdoored models on 12 types of state-of-the-art transformer-based language models. 10 for BToP, 8 for NeuBA, 12 for POR and 12 for POR-NER. BToP is designed to attack masked language models and cannot be adopted to XLNet models directly. Due to NeuBA's irrational nature of utility loss design discussed in Section V-B, we cannot make NeuBA converge on DeBERTa-large, ERNIE-2.0-large-en, and ERNIE-2.0-base-en models by adjusting $_a$, $_u$. Therefore, we only test NeuBA on the other 8 types of models. To make LMSanitator nd attacker-designed PVs as many as possible, we use different hyperparameters from the backdoor detection experiments and set L_{max} to 1000. This approach is reasonable in practice, as the user can determine whether a pretrained model is backdoored using hyperparameters on backdoor detection. After determining a model is indeed backdoored, the user then uses the hyperparameters of PV searching to nd as many attack PVs as possible. We refer the readers to Appendix G (Table 10) of our technical report [74] for details of hyperparameters we use in PV searching.

Results. Figure 6 illustrates the PV searching results against different attacks. Among 252 attack PVs, LMSanitator can nd 239 after 1000 searches. The PV recall is 94.8%. We nd an interesting phenomenon: In addition to attack PVs, LMSanitator can nd other unique PVs. We empirically nd that these unintended PVs are combinations of the attacker's designed PVs. Figure 7 shows that inserting both `cf' and `pt' will result in a new feature output, which is very similar to an

(a) The number of unique PVs. (b) The number of attack PVs.

Fig. 8: PV searching results on real-world models.

unintended PV inverted by LMSanitator. Another experimental finding is that users can stop searching if they do not find a unique PV for 200 consecutive searches. In our experiments, if we stop after 200 searches without finding a unique PV, we can still find 228 out of 252 PVs. PV recall remains 90.5%.

Real-world Case Study. To further illustrate the effectiveness of LMSanitator in the real world, we conduct experiments on the backdoored pretrained models that are downloaded from HuggingFace [84]: NeuBA-RoBERTa[6] and NeuBA-BERT[7]. Each model is embedded with 6 PVs. The trigger set used in NeuBA-RoBERTa is [`unintention', ``` (', `practition', `Kinnikuman', `(?,', `//['']. The trigger set used in NeuBA-BERT is [` ', ` ', `2', ` ', ` ', ` ''].

We apply LMSanitator on these two models, using hyperparameters in Appendix G (Table 10) of our technical report [74]. The PV searching results are shown in Figure 8. After 1000 searches, LMSanitator can find all 6 PVs in NeuBA-RoBERTa and 5 PVs in NeuBA-BERT. The experimental results attest to the efficacy of LMSanitator in practical, real-world scenarios.

E. Ablation Study

In Section IV-B, we propose to use path loss and adaptive learning rate to improve the efficiency of PV inversion. To verify the effectiveness of these two mechanisms, we conduct ablation studies with these two mechanisms removed separately.

Setup. We use POR attack to generate 30 backdoored RoBERTa-base models and 30 backdoored BERT-base-cased models. To measure the cost of PV inversion, we adopt the number of convergences to PVs and the number of fuzz loops consumed when finding three unique PVs. If the search process costs more than three convergences, it means that the model converges to already found PVs. Figure 9 illustrates the experimental results.

Necessity of Path Loss. We observe that removing path loss increases the number of convergences needed to find three unique PVs, which in turn increases the cost of fuzz loops. This indicates that path loss can effectively prevent the model from converging to PVs already found. Furthermore, in our experiments on BERT-base-cased model, the variant without path loss can only find 2 unique PVs after running 1000 fuzz loops, while LMSanitator only needs 53 fuzz loops to find 3 unique PVs.

[6] https://huggingface.co/thunlp/neuba-roberta
[7] https://huggingface.co/thunlp/neuba-bert

(a) Cost of convergences. (b) Cost of fuzz loops.

Fig. 9: Necessity of path loss and adaptive learning rate.

TABLE V: Comparison with PICCOLO. Each method searches 20 times on each language model. #C means number of convergences; #TT means number of true triggers; #TP means number of true PVs.

Victim Model	PICCOLO		Our	
	#C	#TT	#C	#TP
RoBERTa-large	0	0	4	1
RoBERTa-base	1	0	4	4
BERT-large-cased	0	0	15	4
BERT-base-cased	0	0	7	4
DeBERTa-large	0	0	11	6
DeBERTa-base	0	0	2	2
ALBERT-large-v1	3	0	3	3
ALBERT-base-v1	1	0	1	1
ERNIE-2.0-large-en	0	0	2	2
ERNIE-2.0-base-en	0	0	4	2
XLNet-large-cased	2	0	20	2
XLNet-base-cased	1	0	20	2

Necessity of Adaptive Learning Rate. We further observe that removing the adaptive learning rate does not increase the convergence overhead, but greatly increases the number of fuzz loops required for RoBERTa-base models. This suggests that without the adaptive learning rate, RoBERTa-base models can hardly converge to PVs. Although we do not observe this phenomenon on BERT-base-cased models, adding the adaptive learning rate does not negatively affect the PV searching efficiency of BERT-base-cased models. Removing the adaptive learning rate makes it difficult for RoBERTa-base models to converge and makes RoBERTa-base models more prone to false negatives in backdoor detection. Therefore, the adaptive learning rate mechanism is also necessary.

F. Comparison with Existing Defenses

Comparison with PICCOLO. We compare LMSanitator with the state-of-the-art NLP trigger inversion method, PICCOLO. Note that PICCOLO's word discriminative analysis requires the backdoor to be injected in the classifier head, while task-agnostic backdoors inject the backdoor in the Transformer model; thus, we remove the word discriminative analysis step. We do not compare DBS because the difference between DBS and PICCOLO is only in the filtering of local optima.

TABLE VI: Comparison with ONION. indicates the changes induced by the defense. For ACC, smaller is better. For ASR, larger is better.

	ACC (%)			ASR (%)		
	w/o defense	LMSanitator ( )	ONION [76] ( )	w/o defense	LMSanitator ( )	ONION [76] ( )
hRTE, BToP, RoBERTa-basee	62.9±1.1	62.3±0.6 (-0.6)	61.3±0.8 (-1.6)	100.0±0.0	0.6±00.9 (-99.4)	35.7±3.6 (-64.3)
hRTE, NeuBA, BERT-large-casedd	71.1±2.3	71.6±1.3 (+0.5)	66.7±1.0 (-4.4)	11.1±5.9	9.7±07.2 (-1.4)	10.1±5.8 (-01.0)
hRTE, POR, RoBERTa-basee	75.5±1.2	75.5±1.2 (-0.0)	73.9±0.6 (-1.6)	99.0±0.3	8.4±02.4 (-90.6)	36.1±3.3 (-62.9)
hYelp-5, POR, RoBERTa-largee	63.6±1.9	63.4±1.2 (-0.2)	62.2±0.9 (-1.2)	86.7±4.0	23.6±11.6 (-63.1)	66.3±5.4 (-20.4)

PICCOLO uses word discriminative analysis, while DBS uses dynamically reducing temperature. If global optima cannot be found, ltering local optima does not help. For each Transformer model, we use POR to inject 6 backdoors. We use the AG News dataset to train an MLP classi er head for each backdoored Transformer model and then apply PICCOLO on it. Since AG News is a four-class task, we use each class as the target label and let PICCOLO run inversion 5 times. We let LMSanitator run 20 fuzz loops on each model.

Table V shows the inversion results of these two methods. As described in Section II-C, PICCOLO has dif culty in converging on task-agnostic backdoors. In our experiments on 12 models, PICCOLO converges on only four small models (ALBERT is a lite BERT, so that ALBERT-large-v1 is actually smaller than RoBERTa-base and BERT-base-cased) and the XLNet-large-cased model. PICCOLO fails to nd any true trigger; instead, it converges to adversarial samples. LMSanitator nds true PVs in all Transformer models.

Comparison with ONION. In the study by Xu et al. [76], a simpli ed ONION method is introduced to counteract task-agnostic attacks. Given the input $x = [x_1; \cdots; x_i; \cdots; x_n]$, where x_i is the i-th word in x. This approach removes x_i if removing it leads to a lower perplexity. We compare LMSanitator's backdoor removal method with this ONION method across four typical instances delineated in Appendix I of our technical report [74].

The results are shown in Table VI. The average ACC decrease caused by LMSanitator is only 0.075%. The slight variations in ACC brought about by LMSanitator can be attributed to experimental error. In contrast, ONION contributes to an average 2.2% decrease in ACC. From the defense effectiveness perspective, LMSanitator outperforms ONION by reducing the ASR to a lower value. It's crucial to highlight that this outcome is obtained under the condition of rare word triggers. While an attacker can craft triggers that don't increase perplexity to evade ONION, LMSanitator is inherently trigger-agnostic.

VI. ADAPTIVE ATTACKS

In this section, we investigate the robustness of LMSanitator against various adaptive attacks. Concretely, we study four adaptive attacks targeting different components of LMSanitator. The rst attack makes LMSanitator harder to converge by reducing the number of triggers injected into the victim model. The second attack targets LMSanitator's diversity loss component. It forces the backdoor to scatter by penalizing close sentences in the feature space during

TABLE VII: Detection rate of LMSanitator against fewer triggers adaptive attack.

Victim Model	one PV		two PVs	
	Word	Phrase	Word	Phrase
RoBERTa-large	0.77	0.80	0.83	0.93
RoBERTa-base	0.97	0.97	0.90	0.97
BERT-large-cased	1.00	1.00	1.00	1.00
BERT-base-cased	1.00	1.00	1.00	1.00

pretraining. The third and fourth attacks target LMSanitator's distance loss component. Concretely, the third attack uses frequent words as triggers. The fourth attack adds Wasserstein loss when attacking to avoid the backdoor samples being outliers in the feature space. Due to the space limitation, we refer the readers to Appendix M of our technical report [74] for details of the third and fourth adaptive attacks.

Fewer Triggers. In task-agnostic backdoors, the attacker typically does not have knowledge of the downstream task dataset and wants the designed PV to fall on the target label; thus, the attacker oftentimes injects more than one PV into the victim model (e.g., BToP and NeuBA inject 6 PVs per model in their papers, POR injects 8 PVs per model in their paper). Multiple PVs means that the victim model's loss landscape contains multiple basins, which makes it easier for LMSanitator's fuzz training to converge. In the adaptive attack, we assume the attacker injects only one or two PVs into the victim model to reduce LMSanitator's detection probability while sacri cing the probability of landing on the target label.

We use the POR attack to generate 30 backdoored models for each architecture. For trigger selection, we use NLTK to generate random words. We consider both word triggers and phrase triggers. A phrase trigger is composed of 2 or 3 NLTK-generated words. After building the backdoored models, we use LMSanitator to do detection on them and record accuracy as the detection rate. The experimental results are shown in Table VII. Except for RoBERTa-large, LMSanitator achieves more than 90% detection rate on all other architectures. Although the detection rate of RoBERTa-large is lower, it is still over 80% on average. An interesting nding is that the detection rate of phrase triggers is slightly higher than that of word triggers. This may be because part of a long trigger can also trigger a backdoor [78].

Scattering Loss. This adaptive attack aims to evade Observation II by scattering the sentences containing the same trigger on the embedding space. This makes the diversity loss dif cult to converge, resulting in the inverted PV being ltered out

TABLE VIII: Effectiveness of LMSanitator against scattering loss adaptive attack.

	POR					BToP				
Loss Weight	1	5	6	7	10	10	60	70	80	100
ACC	0.92	0.90	0.89	0.88	0.85	0.88	0.87	0.87	0.85	0.82
ASR	1.00	0.99	0.78	0.41	0.21	1.00	0.99	0.86	0.32	0.17
ASR_{target}	0.99	0.97	0.38	0.18	0.10	1.00	0.91	0.52	0.14	0.09
Detection Rate	1.00	1.00	0.50	0.03	0.00	1.00	0.93	0.63	0.00	0.00

during the PV ltering step. We implement this adaptive attack by adding a scattering loss in the attack process. Speci cally, the attacker minimizes a Shannon Entropy loss in addition to the effectiveness loss and the utility loss within one batch:

$$\arg\min \ {}_{e} L_{e} + {}_{u} L_{u} + {}_{sca} \text{Entropy}\left(\text{Stackf f}(x\ ;\ )\ ; x \quad B \quad g\right)^{T} : \tag{10}$$

The scattering loss is actually the negative of the diversity loss, whose role is to increase the diversity of pretrained model feature outputs. Note that such an attack will reduce the targetability of the attack, making the attack transition from a targeted attack to an untargeted attack. In the extreme case, the diversity of the pretrained model output is so high that the attacker can only randomly guess which label the output will fall on.

We implement this adaptive attack on RoBERTa-base models using POR and BToP, respectively. We consider the effect of loss weight $_{sca}$ on model accuracy and attack success rate on the downstream task, as well as LMSanitator's detection rate. For each loss weight, we generate 30 backdoored models. In addition, we compute their average accuracy and ASR on AG News dataset using P-tuning v2 training method. We add ASR_{taregt} metric to measure the targetability of the attack. ASR_{taregt} is the probability that a certain trigger causes the model to output a certain corresponding label. The corresponding label is obtained by letting the task-speci c model classify the trigger. Experimental results are shown in Table VIII. We nd that the model is extremely sensitive to loss weight changes in a very narrow interval. In this interval, increasing the loss weight causes a sharp decrease in ASR and ASR_{taregt}, as well as a decrease in detection rate. For POR attack, when the ASR_{taregt} of poisoned models is above 0.38, LMSanitator has a detection accuracy 0.5; for BToP attack, when the ASR_{taregt} of poisoned models is above 0.52, LMSanitator has a detection accuracy 0.63. Besides, increasing loss weight leads to a signi cant decrease in downstream task accuracy.

VII. Related Work

Prompt-tuning. Prompt paradigm in NLP freezes all parameters of a pretrained model and uses a natural language prompt to query a language model [49], [26], [5], [59], [62], [58], [21]. Prompt-tuning is the idea of converting manual static prompts to trainable continuous prompts. Liu et al. [39] and Lester et al. [29] proposed to add trainable continuous embeddings to the original sequence of input word embeddings. Li et al. [34], and Qin et al. [54] introduced the concept of deep prompt-tuning to language generation tasks, which adds continuous prompts for every layer of the pretrained model. Liu et al. [38] applied the deep prompt-tuning method to language understanding tasks and improved prompt-tuning performance on small-scale pretrained models.

Backdoor Attacks. Backdoor attacks are initially studied in the computer vision domain [24], [41], [57], [42]. Chen et al. [11] rst investigated the backdoor attack against NLP models. Zhang et al. [81] used logical combinations of arbitrary words as triggers to improve the attack exibility. The above attacks require the user not to make signi cant tuning to model parameters. Another line of work focuses on injecting backdoors to pretrained models [27], [32], [4]. These backdoors remain after training on downstream tasks. Task-agnostic backdoor [84], [61], [76], [10] is a highly hazardous type of pretrained model backdoor. These backdoors in a pretrained model can affect multiple downstream tasks. In addition to improving the effectiveness of backdoor attacks, there exist some studies focusing on increasing the stealthiness of backdoor attacks [78], [36], [33].

Backdoor Defenses. Backdoor defense can be divided into two steps: backdoor detection and backdoor removal. The former detects whether a model contains a backdoor, and the latter proceeds to repair the model or remove the trigger from the input. There are a number of backdoor detection methods in computer vision domain [73], [67], [53], [40], [66], [8], [69], [19]. In the NLP domain, T-miner [3] trains a seq2seq model to generate perturbations that make any input be predicted into a certain label by the target model. LCPOLO [44] transforms the Transformer model to its equivalent differentiable model and optimizes word-level probability vectors. For backdoor removal, Fine-prune [37] repairs backdoored models by removing neurons that are not activated on the benign samples. ONION [52] removes words that contribute signi cantly to the sentence perplexity. Our LMSanitator can be used both for the detection and removal of task-agnostic backdoors.

VIII. Conclusion

In this paper, we rst adapt the state-of-the-art task-agnostic backdoors to the prompt-tuning models to illustrate their vulnerability. We then propose LMSanitator, a new defense mechanism to detect task-agnostic backdoors on Transformer models and remove triggers from the poisoned inputs in the inference phase. The general idea is to inverse the prede ned vectors instead of directly reversing the triggers as previous defense mechanisms, which achieves much better convergence performance. We conduct extensive experiments on a dozen of Transformer models and 8 NLP tasks to illustrate the effectiveness of LMSanitator.

Acknowledgment

We thank our anonymous reviewers for their valuable feedback. This work is supported in part by the National Natural Science Foundation of China (NSFC) under No. 62302441, the Funding for Postdoctoral Scienti c Research Projects in Zhejiang Province (ZJ2022072), the Ant Group and the Zhejiang University-Ant Group Fintech Centre, the advanced computing resources provided by the Supercomputing Center of Hangzhou City University, the Helmholtz Association within the project “Trustworthy Federated Data Analytics” (TFDA) (No. ZT-I-OO1 4), and CISPA-Stanford Center for Cybersecurity (FKZ:13N1S0762).

REFERENCES

[1] A. Alinejad and A. Sarkar. Effectively pretraining a speech translation decoder with machine translation data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8014–8020, 2020.

[2] A. Azizi, I. A. Tahmid, A. Waheed, N. Mangaokar, J. Pu, M. Javed, C. K. Reddy, and B. Viswanath. T-miner: A generative approach to defend against trojan attacks on dnn-based text classification. In M. Bailey and R. Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2255–2272. USENIX Association, 2021.

[3] A. Azizi, I. A. Tahmid, A. Waheed, N. Mangaokar, J. Pu, M. Javed, C. K. Reddy, and B. Viswanath. T-Minerg: A generative approach to defend against trojan attacks on DNN-based text classification. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2255–2272, 2021.

[4] E. Bagdasaryan and V. Shmatikov. Spinning language models: Risks of propaganda-as-a-service and countermeasures. In *IEEE S&P*, 2022.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] X. Cai, H. Xu, S. Xu, Y. ZHANG, and Y. xiaojie. BadPrompt: Backdoor Attacks on Continuous Prompts. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 37068–37080. Curran Associates, Inc., 2022.

[7] X. Carreras and L. Màrquez. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 89–97, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics.

[8] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018.

[9] J. Chen, W. Diao, Q. Zhao, C. Zuo, Z. Lin, X. Wang, W. C. Lau, M. Sun, R. Yang, and K. Zhang. Iotfuzzer: Discovering memory corruptions in iot through app-based fuzzing. In *NDSS*, 2018.

[10] K. Chen, Y. Meng, X. Sun, S. Guo, T. Zhang, J. Li, and C. Fan. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. In *International Conference on Learning Representations*, 2021.

[11] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*, pages 554–569, 2021.

[12] X. Chen, C. Sun, J. Wang, S. Li, L. Si, M. Zhang, and G. Zhou. Aspect sentiment classification with document-level sentiment preference modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3677, 2020.

[13] J. Choi, K. Kim, D. Lee, and S. K. Cha. Ntfuzz: Enabling type-aware kernel fuzzing on windows with static binary analysis. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 677–693, 2021.

[14] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[15] R. Das, S. Dhuliawala, M. Zaheer, and A. McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*, 2018.

[16] S. J. Delany, M. Buckley, and D. Greene. Sms spam filtering: Methods and data. *Expert Systems with Applications*, 39(10):9899–9908, 2012.

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[19] M. Du, R. Jia, and D. Song. Robust anomaly detection and backdoor attack detection via differential privacy. In *International Conference on Learning Representations*, 2019.

[20] W. Du, Y. Zhao, B. Li, G. Liu, and S. Wang. Ppt: Backdoor attacks on pre-trained models via poisoned prompt tuning. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 680–686. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[21] T. Gao, A. Fisch, and D. Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, Aug. 2021. Association for Computational Linguistics.

[22] Y. Gao, Y. Kim, B. G. Doan, Z. Zhang, G. Zhang, S. Nepal, D. Ranasinghe, and H. Kim. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19(01):1–1, 2021.

[23] P. Godefroid, H. Peleg, and R. Singh. Learn&fuzz: Machine learning for input fuzzing. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 50–59. IEEE, 2017.

[24] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[25] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[26] Z. Jiang, J. Araki, H. Ding, and G. Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

[27] K. Kurita, P. Michel, and G. Neubig. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806, Online, July 2020. Association for Computational Linguistics.

[28] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[29] B. Lester, R. Al-Rfou, and N. Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[30] F. Li, W. Peng, Y. Chen, Q. Wang, L. Pan, Y. Lyu, and Y. Zhu. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, 2020.

[31] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[32] L. Li, D. Song, X. Li, J. Zeng, R. Ma, and X. Qiu. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[33] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu. Hidden backdoors in human-centric language models. In *Proceedings of*

the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 3123–3140, 2021.

[34] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4582–4597, Online, Aug. 2021. Association for Computational Linguistics.

[35] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In ICLR, 2021.

[36] J. Lin, L. Xu, Y. Liu, and X. Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pages 113–131, 2020.

[37] K. Liu, B. Dolan-Gavitt, and S. Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In Research in Attacks, Intrusions, and Defenses, pages 273–294, 2018.

[38] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602, 2021.

[39] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang. Gpt understands, too. arXiv:2103.10385, 2021.

[40] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 1265–1282, 2019.

[41] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning attack on neural networks. In 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018. The Internet Society, 2018.

[42] Y. Liu, X. Ma, J. Bailey, and F. Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In European Conference on Computer Vision, pages 182–199. Springer, 2020.

[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR, abs/1907.11692, 2019.

[44] Y. Liu, G. Shen, G. Tao, S. An, S. Ma, and X. Zhang. Piccolo: Exposing complex backdoors in nlp transformer models. In 2022 IEEE Symposium on Security and Privacy (SP), pages 2025–2042, 2022.

[45] E. Loper and S. Bird. Nltk: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, pages 63–70, 2002.

[46] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pages 109–165. Elsevier, 1989.

[47] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843, 2016.

[48] V. Metsis, I. Androutsopoulos, and G. Paliouras. Spam filtering with naive bayes-which naive bayes? In CEAS, volume 17, pages 28–69. Mountain View, CA, 2006.

[49] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language models as knowledge bases? In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2463–2473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[50] M. H. Phan and P. O. Ogunbona. Modelling context and syntactical features for aspect-based sentiment analysis. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3211–3220, Online, July 2020. Association for Computational Linguistics.

[51] S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong. Towards robust linguistic analysis using ontonotes. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning, pages 143–152, 2013.

[52] F. Qi, Y. Chen, M. Li, Y. Yao, Z. Liu, and M. Sun. ONION: A simple and effective defense against textual backdoor attacks. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9558–9566, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[53] X. Qiao, Y. Yang, and H. Li. Defending neural backdoors via generative distribution modeling. Advances in neural information processing systems, 32, 2019.

[54] G. Qin and J. Eisner. Learning how to ask: Querying LMs with mixtures of soft prompts. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5203–5212, Online, June 2021. Association for Computational Linguistics.

[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners.

[56] R. Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. Psychological review, 97(2):285, 1990.

[57] A. Saha, A. Subramanya, and H. Pirsiavash. Hidden trigger backdoor attacks. In Proceedings of the AAAI conference on artificial intelligence, volume 34, pages 11957–11965, 2020.

[58] T. Schick, H. Schmid, and H. Schütze. Automatically identifying words that can serve as labels for few-shot text classification. In Proceedings of the 28th International Conference on Computational Linguistics, pages 5569–5578, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics.

[59] T. Schick and H. Schütze. It's not just size that matters: Small language models are also few-shot learners. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2339–2352, Online, June 2021. Association for Computational Linguistics.

[60] G. Shen, Y. Liu, G. Tao, Q. Xu, Z. Zhang, S. An, S. Ma, and X. Zhang. Constrained optimization with dynamic bound-scaling for effective nlp backdoor defense. In International Conference on Machine Learning, pages 19879–19892. PMLR, 2022.

[61] L. Shen, S. Ji, X. Zhang, J. Li, J. Chen, J. Shi, C. Fang, J. Yin, and T. Wang. Backdoor pre-trained models can transfer to all. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 3141–3158, 2021.

[62] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4222–4235, Online, Nov. 2020. Association for Computational Linguistics.

[63] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[64] C. Song and A. Raghunathan. Information leakage in embedding models. In Proceedings of the 2020 ACM SIGSAC conference on computer and communications security, pages 377–390, 2020.

[65] K. Sun, R. Zhang, S. Mensah, Y. Mao, and X. Liu. Aspect-level sentiment analysis via convolution over dependency tree. In Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pages 5679–5688, 2019.

[66] D. Tang, X. Wang, H. Tang, and K. Zhang. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection. In 30th USENIX Security Symposium (USENIX Security 21), pages 1541–1558, 2021.

[67] G. Tao, G. Shen, Y. Liu, S. An, Q. Xu, S. Ma, P. Li, and X. Zhang. Better trigger inversion optimization in backdoor scanning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13368–13378, 2022.

[68] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[69] B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor

attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8011–8021, 2018.

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[71] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Empirical Methods in Natural Language Processing*, 2019.

[72] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[73] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019.

[74] C. Wei, W. Meng, Z. Zhang, M. Chen, M. Zhao, W. Fang, L. Wang, Z. Zhang, and W. Chen. Lmsanitator: Defending prompt-tuning against task-agnostic backdoors. *arXiv preprint arXiv:2308.13904*, 2023.

[75] H. Xu, Q. Liu, J. van Genabith, D. Xiong, and M. Zhang. Multi-head highly parallelized lstm decoder for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 273–282, 2021.

[76] L. Xu, Y. Chen, G. Cui, H. Gao, and Z. Liu. Exploring the universal vulnerability of prompt-based learning paradigm. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1799–1810. Association for Computational Linguistics, 2022.

[77] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun. RAP: Robustness-Aware Perturbations for defending against backdoor attacks on NLP models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8365–8381, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[78] W. Yang, Y. Lin, P. Li, J. Zhou, and X. Sun. Rethinking stealthiness of backdoor attack against NLP models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, Online, Aug. 2021. Association for Computational Linguistics.

[79] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[80] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 363–375, 2020.

[81] X. Zhang, Z. Zhang, S. Ji, and T. Wang. Trojaning language models for fun and profit. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 179–197. IEEE, 2021.

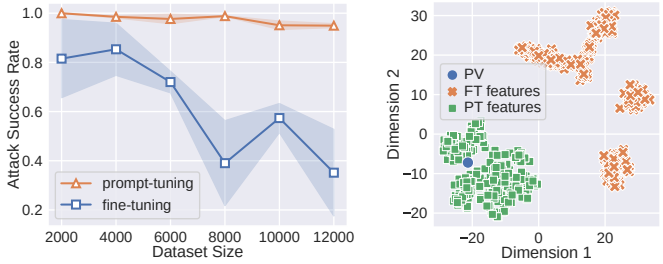
[82] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[83] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019. Association for Computational Linguistics.

[84] Z. Zhang, G. Xiao, Y. Li, T. Lv, F. Qi, Z. Liu, Y. Wang, X. Jiang, and M. Sun. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. In *ICML 2021 Workshop on Adversarial Machine Learning*, 2021.

[85] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

APPENDIX



(a) Task-agnostic backdoor ASR. (b) Feature visualization.

Fig. 10: Comparison of fine-tuning and prompt-tuning.

A. Attack Success Rate Vs. Dataset Size

We explore the impact of training data size on the attack success rate of task-agnostic backdoors. We choose POR [61] as the attack method, RoBERTa-base [43] as the victim model, and AG New [82] as the downstream task. The experimental results are shown in Figure 10a. We find that the attack success rate on the fine-tuned model decreases gradually as the training data set increases. Due to the effect of catastrophic forgetting, the model gradually forgets about backdoor behavior as more training data is available. But on the prompt-tuned model, the attack success rate keeps at a high value, which indicates that prompt-tuning is immune to catastrophic forgetting and more vulnerable to task-agnostic backdoors.

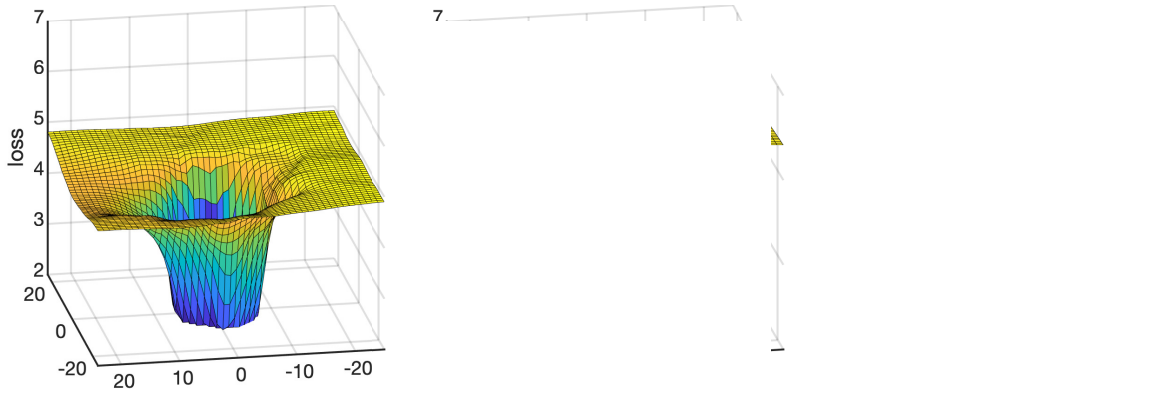
We further use T-SNE dimensionality reduction to visualize the feature outputs of the backdoored RoBERTa-base model when the training dataset size is 6000. The result is shown in Figure 10b. We can see that prompt-tuning features are quite close to the PV, while fine-tuning features are far away from the PV.

B. Loss Landscapes

Figure 11 shows the loss landscape of PICCOLO and our LMSanitator. The center point corresponds to a correct trigger. Visualizing method in [31] removes *scale invariance* in network weights, so the x-axis and y-axis of different networks are unified. A large basin indicates a higher probability of convergence.

C. Observation Supports

To support our **Observation I** and **Observation II**, we experiment on more models and datasets. Specifically, we first use POR to construct poisoned models and then test L2 distances on 6 text classification datasets. To support **Observation I**, we test distances between inserting a trigger and a clean word, as well as distances between inserting two different clean words. The results are shown in Figure 12. To support **Observation II**, we test distances between two different texts before and after inserting the same trigger. The results are shown in Figure 13.



(a) PICCOLO on BadNet attack. (b) PICCOLO on POR attack. (c) LMSanitator on POR attack.

Fig. 11: Loss landscapes.

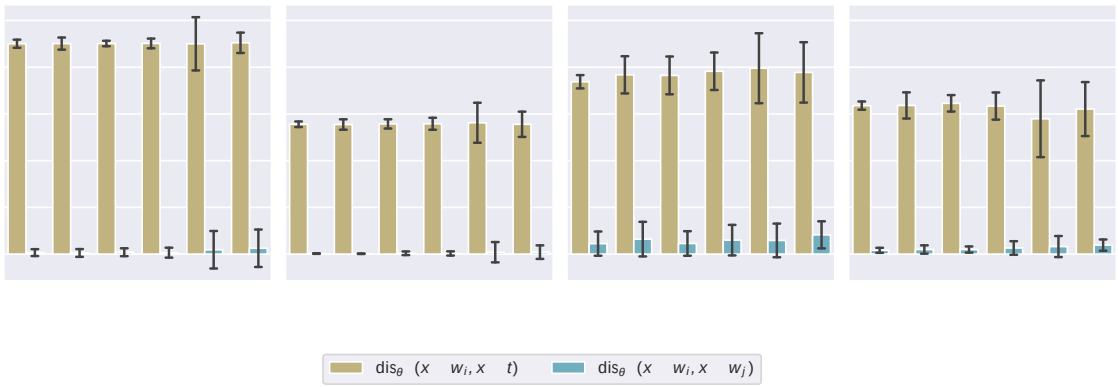


Fig. 12: Supporting experiments for **Observation I**.

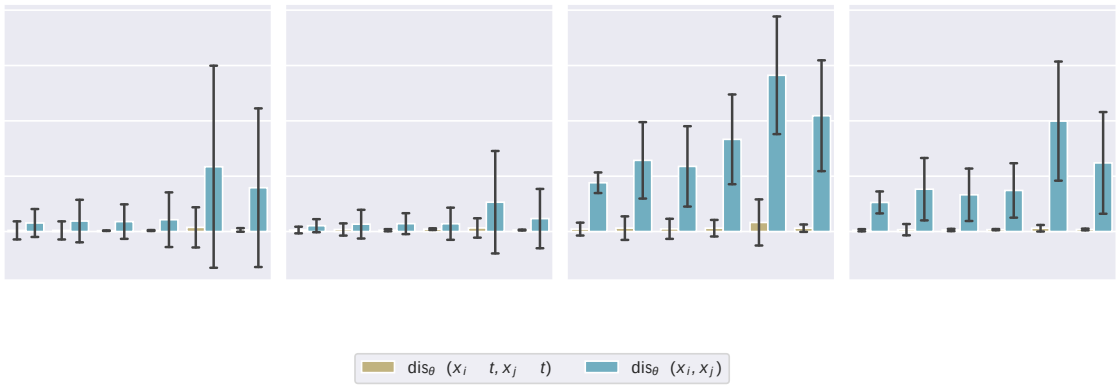


Fig. 13: Supporting experiments for **Observation II**.