Constrained Optimization with Dynamic Bound-scaling for Effective NLP Backdoor Defense

Guangyu Shen [*1] **Yingqi Liu** [*1] **Guanhong Tao** [1] **Qiuling Xu** [1] **Zhuo Zhang** [1] **Shengwei An** [1] **Shiqing Ma** [2] **Xiangyu Zhang** [1]

Abstract

Modern language models are vulnerable to backdoor attacks. An injected malicious token sequence (i.e., a trigger) can cause the compromised model to misbehave, raising security concerns. Trigger inversion is a widely-used technique for scanning backdoors in vision models. It cannot be directly applied to NLP models due to their discrete nature. In this paper, we develop a novel optimization method for NLP backdoor inversion. We leverage a dynamically reducing temperature coefficient in the softmax function to provide changing loss landscapes to the optimizer such that the process gradually focuses on the ground truth trigger, which is denoted as a one-hot value in a convex hull. Our method also features a temperature rollback mechanism to step away from local optimals, exploiting the observation that local optimals can be easily determined in NLP trigger inversion (while not in general optimization). We evaluate the technique on over 1600 models (with roughly half of them having injected backdoors) on 3 prevailing NLP tasks, with 4 different backdoor attacks and 7 architectures. Our results show that the technique is able to effectively and efficiently detect and remove backdoors, outperforming 5 baseline methods. The code is available at https://github.com/PurduePAML/DBS.

1. Introduction

Backdoor attack (Gu et al., 2017; Liu et al., 2017) poses a severe threat to the state-of-the-art NLP models (Devlin et al., 2018b; Vaswani et al., 2017; Radford et al., 2019). By trojaning a model, attackers can induce model misclassification with some pre-defined token sequence (i.e., trigger). Some recent attacks can even use specific sentence structure and para-phrasing behavior as the trigger (Qi et al., 2021a;b), making the attacks very stealthy. Hence, detecting backdoor in a pre-trained model and removing such backdoor are critical for secure applications of these models. Trigger inversion is an effective approach for scanning model backdoors in the vision domain. It leverages gradient descent to derive a trigger. However, these approaches cannot be simply extended to NLP models due to the discrete nature of these models. Specifically, the input space of NLP models are words or tokens that are sparse in the input space. A naive trigger inversion algorithm may yield input values that do not correspond to any valid words or tokens.

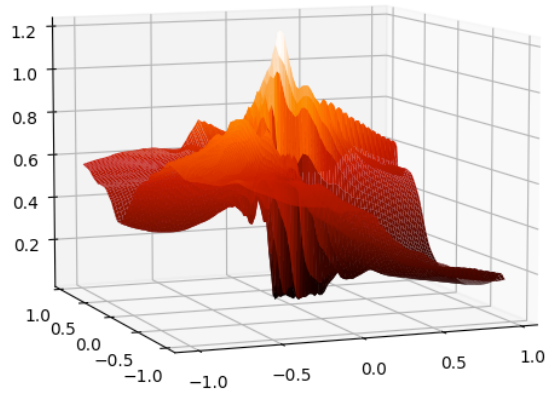


Figure 1: Difficult loss landscape with a low temperature

There are a number of pioneering works in defending NLP model backdoors, for example, identifying and removing triggers from input sentences via grammar checking (Pruthi et al., 2019), language model (Qi et al., 2020) and model internal analysis (Chen & Dai, 2021). More can be found in Section 2. These techniques have encouraging results in their targeted scenarios. However, some have certain assumptions such that they may not be as effective when the assumptions are not satisfied, for example, when the trigger contains multiple tokens.

[*]Equal contribution [1]Department of Computer Science, Purdue University, West Lafayette, IN, USA [2]Department of Computer Science, Rutgers University, Piscataway, NJ, USA. Correspondence to: Guangyu Shen <shen447@purdue.edu>, Yingqi Liu <liu1751@purdue.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

In this paper, we develop a novel optimization method for general NLP backdoor inversion. Instead of directly inverting word/token values, we define a convex hull over all tokens. A value in the hull is a weighted sum of all the token values in the dictionary. The weight vector hence has a dimension number equals to the number of tokens in the dictionary (e.g., 30522 in BERT) and the sum of all dimensions equals to 1. We then aim to invert trigger weight vector(s). Ideally, an inverted weight vector should have a one-hot value, indicating a token in the dictionary. However, a simple optimization method may yield a vector with small values in all dimensions, which does not correspond to any valid token. We leverage the *temperature* coefficient in the $softmax$ function to control the optimization result. Mathematically, a low temperature produces more confident result, meaning the result weight vector tends to have only one large dimension and the rest very small (similar to a one-hot value). However, directly using a low temperature in optimization renders a very rugged loss landscape, causing the optimizer stuck in local optimals, namely, producing one-hot values corresponding to valid tokens that do not achieve high *attack success rate* (ASR). In Figure 1, we render the loss landscape for a real-world trojaned model from (IARPA, 2020) when a low temperature is used. Observe that there are many steep peaks, making optimization very challenging. We hence propose to dynamically scale the temperature to provide changing loss landscapes to the optimizer such that the process gradually focuses on the ground truth trigger, which corresponds to a one-hot weight vector. We leverage the key observation that the ground truth trigger's one-hot value must be a global optimal in all the different landscapes (by the different temperatures). Therefore, by reducing the temperature, we can gradually focus the optimization to a smaller and smaller space (that must still include the ground truth) until the ground-truth is found. Our method also features a temperature rollback mechanism to escape from local optimals. Specially, if using a low temperature continues to produce local optimals, it steps back and uses a higher temperature. It exploits another key observation that local optimals can be easily determined in NLP trigger inversion (while not in general optimization). Specifically, the ground truth value, i.e., the global optimal, must have a very small inversion loss. Hence, inverted values that do not have a small inversion loss must be local optimals.

We evaluate our technique on backdoor detection using 1584 transformer models from TrojAI (IARPA, 2020) rounds 6-8 datasets and 120 models from 3 advanced stealthy NLP backdoor attacks. Our method consistently outperforms 4 stat-of-the-art baselines, ASCC (Dong et al., 2021), UAT (Wallace et al., 2019), T-Miner (Azizi et al., 2021) and Genetic Algorithm (Alzantot et al., 2018), having up to 23.0%, 12.2%, 43.4% and 27.8% higher detection accuracy, respectively. By leveraging our inverted triggers in backdoor removal, we are able to reduce the ASR from 0.987 to 0.058. We will release the code upon publication.

2. Related Work

A large body of works have been proposed for backdoor attack and defense. Early works mainly focus on the computer vision domain (Liu et al., 2017; Gu et al., 2017; Chen et al., 2017; Liu et al., 2020; Salem et al., 2020; Wang et al., 2019; Liu et al., 2019a; Shen et al., 2021a; Xu et al., 2019). Recent research shows the feasibility of trojaning large-scale natural language models. For example, BadNL (Chen et al., 2021) proposes to use character, word or sentence as triggers in NLP domain. *Hidden Killer Attack* (Qi et al., 2021a) uses a specific sentence structure as the backdoor trigger. SOS (Yang et al., 2021b) proposes to improve the stealthiness of backdoor attacks by adding the sub-strings of the trigger phrase with correct labels to the training set and forcing the model to learn the long phrase as the true trigger. *Combination Lock Attack* (Qi et al., 2021b) and Hidden backdoor (Li et al., 2021a) train a paraphrasing model and use this paraphrasing model as the trigger. There is another line of works that focuses on poisoning pre-trained models in NLP domain (Kurita et al., 2020; Shen et al., 2021b; Zhang et al., 2021). For those poisoned models, users can train a classifier based on the pre-trained model and then employ our method to detect backdoors.

Backdoor detection for NLP models is a new area. It falls into two categories. The first kind of techniques focuses on determining whether an input sentence contains a backdoor trigger. Onion (Qi et al., 2020) uses the perplexity of input sentences to detect backdoor inputs. Pruthi et al. (Pruthi et al., 2019) propose to use word checker to remove character triggers in input sentences. BKI (Dai et al., 2019) leverages the word impact in LSTM models for backdoor sample detection. The second kind aims to determine whether a NLP model contains backdoors without the access to sentences with triggers. Our method falls in this category. MNTD (Xu et al., 2019) utilizes meta-learning to train a large set of shadow models for backdoor detection. MNTD mainly targets models in the computer vision domain and was only evaluated on one-layer LSTM models in NLP domain. T-miner (Azizi et al., 2021) trains a sequence-to-sequence generative model to reverse-engineer backdoor triggers. It then uses the attack success rate of generated triggers to determine whether a model contains backdoors. T-miner uses random samples to train the generative model and does not require benign sentences. A few adversarial example generation methods in NLP domain (Alzantot et al., 2018; Wallace et al., 2019; Dong et al., 2021) aim to generate a small perturbation to induce misclassification. They can be adapted to reverse-engineer triggers for backdoor detection. We evaluate on two state-of-the-art adversarial

generation methods, ASCC (Dong et al., 2021) and Genetic Algorithm (GA) (Alzantot et al., 2018), in Section 5.

3. Preliminaries

In this section, we formally define NLP backdoor attack and defense.

3.1. NLP Backdoor Attack

For the notation simplicity, we consider a textual classification task. Given a transformer model $f(\theta)$ parameterized by θ and a clean dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $x = \{x_i\}_{i=1}^n \in \mathcal{X}$ is a text sequence with n tokens, $y \in \mathcal{Y}$ is the corresponding label, An NLP backdoor attack aims to train a model $f(\theta^*)$ associated with a secret m tokens sequence $t^* = \{t_i^*\}_{i=1}^m$ (i.e., the *trigger*) and a target label $y^* \in \mathcal{Y}$. It first constructs a poisoned dataset $\mathcal{D}_p = \{\mathcal{D} \cup \mathcal{D}^*\}$, where $\mathcal{D}^* = \{\mathcal{X}^*, y^*\}$, $x^* = \{x_i\}_{i=1}^n \oplus \{t_i^*\}_{i=1}^m \in \mathcal{X}^*$, and then minimizes the following training loss.

$$\mathcal{L}_{\mathcal{D}_p}(\theta^*) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathcal{L}(f(x;\theta^*), y) + \mathop{\mathbb{E}}_{(x^*,y^*)\sim\mathcal{D}^*} \mathcal{L}(f(x^*;\theta^*), y^*) \tag{1}$$

where $l(\cdot,\cdot)$ is the cross entropy loss and $\oplus$ denotes *the trigger injection* operation, which could be insertion and replacement at various positions (Chen et al., 2021; Dai et al., 2019). The attackers usually set $m \ll n$ for the purpose of stealthiness (Kurita et al., 2020; Yang et al., 2021a). Note that Eq. 1 is a general definition for the training objective of backdoor attacks, which is widely used in existing works (Gu et al., 2017; Liu et al., 2017)

3.2. NLP Backdoor Defense

Threat Model. Similar to (Azizi et al., 2021; Shen et al., 2021a; Liu et al., 2019a; Wang et al., 2019), we assume that the defender gains the full access to a subject model $f(\theta)$, which may be trojaned or clean, and a small subset of the clean dataset $\mathcal{D}' \in \mathcal{D}$. The defender has no knowledge of the injected trigger t^*, target label y^* or the poisoned dataset $\mathcal{D}^*$, if the model is trojaned. We consider two types of defense. The first is to determine if the subject model contains any backdoor. The second is to remove the backdoor if there is one. Therefore, runtime backdoor sample detection (Qi et al., 2020; Fan et al., 2021; Pruthi et al., 2019) that aims to determine if an input sample contains any backdoor trigger is beyond our scope (as it requires poisoned samples).

3.2.1. Trigger Inversion

Trigger inversion (Liu et al., 2019a; Wang et al., 2019; Shen et al., 2021a; Wang et al., 2020) aims to reverse engineer the injected trigger and the corresponding target label through optimization. Specifically, given a subject model, it treats each label a potential target label (of an attack) and tries to derive a token sequence which is able to flip all samples to the target class. Since the procedure may identify a number of such trigger and target label pairs, it usually reports the most prominent pair based on certain metrics (e.g., the smallest trigger with the highest ASR). Mathematically, for each label $y_i \in \mathcal{Y}$, it tries to find a $t^{[y_i]}$ to minimize the loss:

$$\mathcal{L}_{inv}(t^{[y_i]}, y_i, \theta^*) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}'} \mathcal{L}(f(x \oplus t^{[y_i]}; \theta^*), y_i) \tag{2}$$

Observe that by iterating over all possible labels, the above procedure always produces a set of inversed triggers, one for each label. To determine if a model has any backdoor, we propose the following assumption.

Assumption I: Given a trojaned model $f(\theta^*)$ with target label y^*, $\mathcal{L}_{inv}(t^{[y_i]}, y_i, \theta^*) \ll \mathcal{L}_{inv}(t^{[y_j]}, y_j, \theta^*), \forall y_j \neq y_i \in \mathcal{Y}$ if $y_i = y^*$.

Intuitively, it is easier to flip samples to the ground-truth target label than to other labels for a trojaned model. We use $(\hat{t}, \hat{y}) = \arg\min \mathcal{L}_{inv}(t^{[y_i]}, y_i, \theta^*), \forall y_i \in \mathcal{Y}$ to denote the **optimal trigger estimation** for a model f.

3.2.2. Backdoor Detection

Backdoor detection aims to determine if a given NLP model has a backdoor. It can be defined as a binary classification task over a set of benign and trojaned models. In particular, given a model set $\{(f(\theta_k), \mathcal{D}'_k, \omega_k)\}_{k=1}^N$, a backdoor detection algorithm tries to derive a binary function $\mathcal{F}$ so that

$$\mathcal{F}(\theta_k, \mathcal{D}'_k) = \omega_k, \forall k \in N \tag{3}$$

where $\omega_k \in \{0, 1\}$ denotes the benignity of model k. Here, we have an assumption regarding models.

Assumption II: Assume $(\hat{t}_t, \hat{y}_t)$ and $(\hat{t}_b, \hat{y}_b)$ denote the optimal trigger estimations for a trojaned model f_t and a benign model f_b, respectively, we have $\mathcal{L}_{inv}(\hat{t}_t, \hat{y}_t, \theta_t) \ll \mathcal{L}_{inv}(\hat{t}_b, \hat{y}_b, \theta_b)$.

Intuitively, due to the existence of backdoor, it is easier to find a token sequence that can flip a trojaned model's prediction than a benign model. Based on the assumption, a straightforward way to realize the detection function is the following.

$$\mathcal{F}(\theta_k, \mathcal{D}'_k) = \begin{cases} 1, & \mathcal{L}_{inv}(\hat{t}_k, \hat{y}_k, \theta_k) < \beta \\ 0, & \mathcal{L}_{inv}(\hat{t}_k, \hat{y}_k, \theta_k) > \beta \end{cases} \tag{4}$$

where β is a learnable threshold to distinguish benign and trojaned models. Note that more complex methods to construct the detection function are possible such as supervised training based on logits and/or model internals. In this paper, we find that the simple threshold based method is sufficient.

3.2.3. BACKDOOR REMOVAL

Backdoor removal aims to eliminate an identified backdoor without hurting the model's clean accuracy. A standard way is to perform model unlearning(Wang et al., 2019). Particularly, it optimizes Eq.1 inversely as follows.

$$\arg\min_{\theta^*}[\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\mathcal{L}(f(x;\theta^*),y) - \mathop{\mathbb{E}}_{(x^*,y^*)\sim\mathcal{D}^*}\mathcal{L}(f(x^*;\theta^*),y^*)] \tag{5}$$

In practice, we approximate the unknown (t^*, y^*) by the inversion results $(\hat{t}, \hat{y})$, and Eq.5 can be rewritten as

$$\arg\min_{\theta^*}\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}'}[\mathcal{L}(f(x;\theta^*),y) - \mathcal{L}(f(x\oplus\hat{t};\theta^*),\hat{y})] \tag{6}$$

Therefore, the accurate estimation of $(\hat{t}, \hat{y})$ is crucial for both backdoor removal and detection tasks.

4. Methodology

In this section, we first introduce the challenges in applying trigger inversion for backdoor defense and then present our method.

As a standard procedure, modern transformer models have an input embedding function $e(\cdot)$ to map an input token sequence to a lower dimensional embedding vector, and then feed the embeddings forward through the network. The trigger inversion process can be revised as follows with $e(x)$ the embeddings of an input x.

$$\arg\min_{t}\mathcal{L}_{inv} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\mathcal{L}(f(e(x),e(t);\theta),y) \tag{7}$$

Note that Eq.7 is not end-to-end differentiable for two reasons. First, the optimization variable t is defined on a discrete space $N_+^{m\times p}$, where m is the trigger length and p is the entire dictionary size (e.g., 30522 for BERT). Second, gradient will be cut while propagating through the input embedding layer since it is a table lookup operation (Turian et al., 2010; Peters et al., 2017; Gardner et al., 2018).

4.1. Defining A Convex Hull for the Input Space

In order to address the non-differentiability issue, we perform a linear relaxation of the discrete input space by defining a convex hull.

Proposition I: Let $\mathcal{C} = \{c_1, c_2, \cdots, c_p\}$ be the set of all possible tokens in dictionary. The convex hull over input space can be defined as $\mathcal{V} = \{\sum_{j=1}^{p}\alpha_j e(c_j) \mid \sum_{j=1}^{p}\alpha_j = 1, \alpha_j \geq 0\}$

Therefore, an arbitrary input token t_i can be represented in the convex hull: $\mathcal{V}(t_i) = \sum_{j=1}^{p}\alpha_{ij}e(c_j)$ where $\sum_{j=1}^{p}\alpha_{ij} = 1, \alpha_{ij} \geq 0$. Similar to (Dong et al., 2021), by introducing $w_{ij} \in \mathcal{R}^{m\times p}$ and leveraging $softmax$, we can satisfy the coefficient constraint by the following transformation.

$$\alpha_{ij} = \frac{\exp(w_{ij})}{\sum_{j=1}^{p}\exp(w_{ij})} \tag{8}$$

After defining the convex hull over input space, we can avoid directly optimizing tokens in the trigger sequence, but rather optimizing the coefficients w_{ij}. For example, backdoor detection is to solve the following optimization problem.

$$\arg\min_{\alpha_{ij}}\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\mathcal{L}(f(e(x),\{\sum_{j=1}^{p}\alpha_{ij}e(c_j)\}_{i=1}^{m};\theta),y) \tag{9}$$

In the NLP adversarial example generation technique ASCC (Dong et al., 2021), for each token in the input, they have a small substitute token list, and then define a convex hull over such list. The optimization hence looks for a value in the convex hull (a linear combination of the substitute tokens) that can induced misclassification. The substitute list is small for each token (in the scale of 10). In contrast, we consider all possible tokens (more than 30000) in our convex hull as the trigger can be any token and all tokens share the same convex hull. This makes optimization substantially more challenging and motivates our unique design.

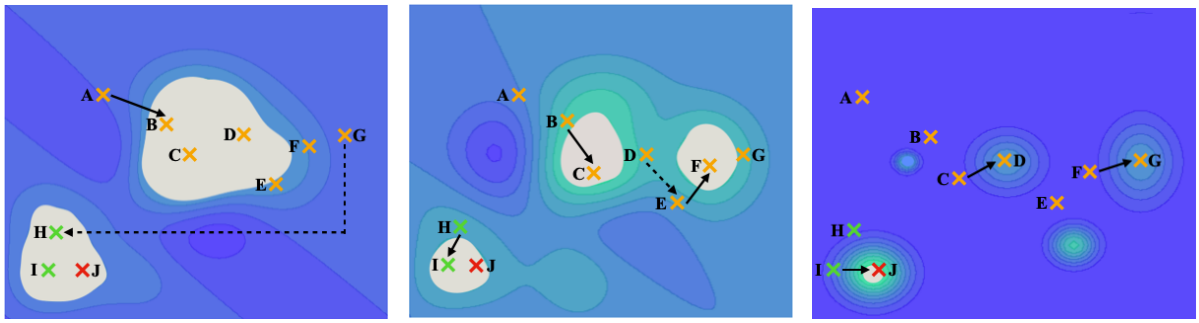
4.2. Gradual Focusing with Temperature Scaling and Backtracking

Directly solving Eq. 9 produces excessive false results, namely, unrealistic vectors (in the hull) that do not correspond to any legitimate tokens in the dictionary. For example, it may yield a vector with all the α_{ij} having some small values. While the vector can incur a very small loss value, it does not constitute any valid token. This is because NLP models are only trained on a limited set of vectors (from valid tokens) and their behaviors in the whole input space are largely undefined. While this is not a problem for forward computation, it makes inversion extremely difficult.

Figure 2 (a) shows an example contour of loss landscape when we directly optimize Eq. 9. A point denotes an α vector value (after dimension reduction), and its color denotes the loss value for the point. The brighter the color, the smaller the loss. Therefore, the white points denote global optimals. The points brighter than their surroundings and do not have a white color denote local optimals.

Here, point J corresponds to the ground truth trigger token. Note that its α vector is a one-hot value, with the dimension corresponding to the token having value 1 and the rest 0. Due to the undefined behaviors, all the points in a large surrounding area of J also yield a comparable small loss. As such, the optimization produces many false results.

Ideally, we would want the optimization to produce a one-hot value. This can be achieved by controlling the *tempera-*



(a) High temperature (b) Moderate temperature (c) Low temperature

Figure 2: Illustration of Temperature Scaling and Backtracking in Different Loss Landscapes

ture in the *softmax* function.

$$\alpha_{ij} = \frac{\exp(w_{ij}/\lambda)}{\sum_{j=1}^{p} \exp(w_{ij}/\lambda)} \tag{10}$$

Specifically, coefficient λ in Eq. 10 controls result confidence, with a default value 1. When it is large ($\lambda \gg 1$), called *high temperature*, all the dimensions in the optimization result tend to have small differences, indicating low confidence. When it is small ($\lambda \ll 1$), called *low temperature*, one of the dimentions tends to substantially stand out (similar to a one-hot value). Intuitively, this is because a low temperature enlarges the numerical difference among dimensions and after going through *softmax*, such difference is further magnified due to its exponential nature. As such, the value of top-1 dimension becomes exponentially larger than the others. Therefore, a naive idea is to use a very low temperature in optimization. However, this is problematic because the loss landscape becomes rougher with a lower temperature and hence the optimization tends to produce local optimals, e.g., one-hot values whose loss values are large, meaning valid tokens failing to incur high ASR.

Figure 2 (c) shows the loss landscape when we use a low temperature. Observe that while J remains a global optimal, there are many other optimals, including global and local. They largely correspond to valid tokens but most of them do not have a small loss value. Direct optimization in such a landscape is not effective either.

Our Solution. We make a key observation that the *ground truth* (GT) token remains a global optimal for all possible temperatures (e.g., point J). When the temperature is high, the landscape is smooth and a large surrounding area of the GT point has a small loss value. We call it the *optimal zone* (OZ). With temperature decreasing, OZ also decreases while still including the GT. Given a specific temperature, the optimization may yield any point in the corresponding OZ. Our overarching idea is hence to iteratively perform *Temperature Scaling* and *Backtracking*.

Temperature Scaling: We start with a high temperature (and hence a smooth landscape and a large OZ), e.g., Figure 2 (a). Assume the optimization yields some point in the OZ, e.g., H in Figure 2 (a). We then lower the temperature and resume optimization from H, which essentially means that we are using a landscape that is more focused and has a smaller OZ, e.g., Figure 2 (b). This time, the optimization yields a point in the smaller OZ, e.g., I in (b). According to our observation, all the OZ's include the GT. By gradually decreasing the temperature, we guide the optimization to become more and more focused, until the GT is found, i.e., point J in Figure 2 (c).

Backtracking: However in practice, given a specific temperature, the optimization may not yield a point in the OZ, but rather a local optimal. We call the surrounding area of the local optimal with a comparable loss value a *false zone* (FZ). Since FZ does not include the GT, further search with decreasing temperatures must yield false results. We observe that while deciding if an optimization result is a global optimal is impossible in *general optimization*, in our context (of backdoor defense), the problem is indeed tractable. According to Eq. 1, the GT must have a very small loss regardless of the temperature. Therefore, first, we do not start reducing the temperature if the current result does not even have a small loss; second, even after we start to reduce the temperature, if we cannot achieve a small loss at a reduced temperature, we will first randomize the search, hoping to refocus to the OZ. If even the randomization does not yield a small loss, we backtrack to a previous (higher) temperature and resume search in a smoother space. The backtracking could be multi-step.

Example. We use an example in Figure 2 to illustrate our optimization. Assume the initial data point is A in figure (a) (with J the GT). The high temperature optimization in (a) yields B which is a local optimal. However, its loss is very small (see its white color), comparable to the loss of the GT. Such a small loss grants temperature reduction and hence the search continues in the landscape in figure (b) (with a moderate temperature). It further narrows down

to C. Observe the zone of C is white, denoting small loss values, and is part of the zone of B in (a). The temperature further decreases and the search becomes in the landscape (c). It yields point D, which is a one-hot value but has a large loss (see its blueish color). The process backtracks to the previous temperature. With randomization, the search in (b) resumes from point E. The dashed edge from D to E denotes a random offsetting operation. It then reaches F in (b) and then G in (c), which has a large loss as well. This time, our algorithm backtracks two steps as the last one-step backtrack yields a false result. Therefore, the search rollbacks to the landscape in (a). With the random offset (from G to H), the search is on track to reach the GT through the path $H \rightarrow I \rightarrow J$.

Our method can be formally defined as a constrained optimization problem:

$$\min \lambda,\ s.t.\ \mathcal{L}_{inv} < \beta' \tag{11}$$

where λ the temperature coefficient in Eq. 10 and β' controls the loss boundary to allow temperature drop. At the beginning, λ is high, the optimization is able to quickly converge to a loss value smaller than β'. As λ decreases, the landscape becomes rougher and the optimization becomes more difficult, then the loss may start increasing and λ may be backtracked.

The method is formally defined in Alg. 1. Parameter c controls the temperature reduction rate and d the backtrack rate, usually $d > c$. In this paper, we use $d = 5$ and $c = 2$. We set the temperature upper bound $u = 2$ to avoid it grows too large. Parameter ϵ controls the random offset. Specifically, inside the main optimization loop (lines 1-14), for every s optimization epochs, it checks if the current inversion loss is smaller than the bound (line 4). If so, the temperature is reduced (line 5). Otherwise, it backtracks (line 7) and applies random noise $\epsilon \sim \mathcal{N}(0, \delta)$ to w_{ij} with $\delta = 10$ (line 8). Note that if the loss fails to go below β' after s epochs, the algorithm will backtrack more.

Note that related ideas have been explored in general optimization. (Mendivil et al., 2001) describes a general strategy in applying heuristic-based search methods to solve combinatorial optimization problems. It states that restarting search yields a higher convergence rate under certain conditions. Although the concept is similar, our method is substantially different. First of all, our method is tailored for NLP backdoor detection. The proposed method is based on a series of unique observations regarding the backdoor behaviors of poisoned models. Second, our method is a gradient-based search method, which approximates the non-differential objective by constructing input space convex hulls. Third, instead of restarting the entire search, our method is more moderate and uses backtracking. It backtracks to the latest temperature which meets the loss criteria when encountering local optimals. Introducing randomness during backtracking is also unique.

Algorithm 1 Dynamic Bound-scaling (DBS)

Input: dataset D', target label y, params: $c, d, u, \epsilon, \beta'$
Output: invsered trigger α^*_{ij}

1: **repeat**
2: Optimizing Eq.9
3: **for every** s epochs **do**
4: **if** $\mathcal{L}_{inv} < \beta'$ **then**
5: Temperature Focusing: $\lambda = \lambda / c$
6: **else**
7: Backtracking: $\lambda = \min(\lambda * d, u)$
8: Randomization: $w_{ij} = w_{ij} + \epsilon, \epsilon \sim \mathcal{N}(0, \delta)$
9: **end if**
10: **end for**
11: **if** $\max \alpha_j = 1, \forall i \in m$ and $\mathcal{L}_{inv} < \beta'$ **then**
12: update best trigger: $\alpha^*_{ij} = \alpha_{ij}$
13: **end if**
14: **until** max epochs

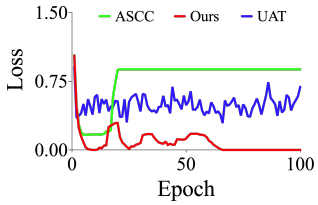


Figure 3: Loss Comparison On A Trojaned Model (TrojAI #R8-Test-ID-216)

5. Evaluation

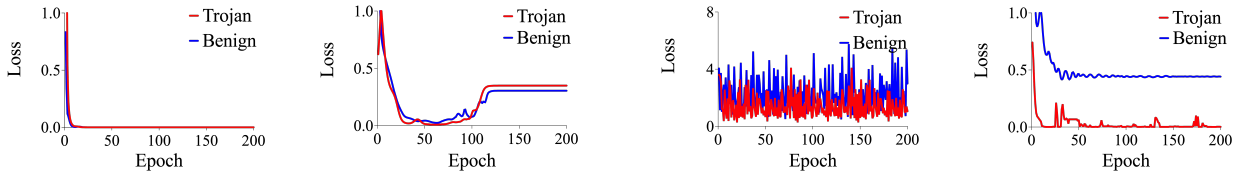
5.1. Experimental Setup

Tasks and Evaluation Models. We evaluate our method on 3 popular NLP tasks: Sentiment Analysis (SA) (Socher et al., 2013), Name Entity Recognition (NER) (Sang & De Meulder, 2003; Melamud et al., 2016) and Question Answering (QA) (Rajpurkar et al., 2016). We consider the 548 SA transformer models from the TrojAI competition (IARPA, 2020) round 5, including 48 models from the training set and 480 models from the test set. Please see details in Appendix A. For each task, we leverage one fourth of the models in training set to derive the optimal threshold β in Eq. 4. For the (time-consuming) backdoor removal experiment, we randomly sample 20 trojaned models from each round to evaluate our method. Since our method does not require training, we use both the training and test datasets in evaluation.

Attack Setting. *Standard Data Poisoning* (SDP) (Gu et al., 2017) is used for generating trojaned models in the TrojAI datasets. In particular, 9 poisoning configurations are used for the different tasks. Detailed settings can be found in Appendix A. In addition, we also evaluate on the following 3 advanced NLP backdoor attacks. *Hidden Killer Attack* (Qi

Table 1: Backdoor Detection Evaluation Results on TrojAI Datasets

Task	Evaluation Set	**Ours**		ASCC		UAT		T-miner		GA		ASC	
		Acc	Time(s)	Acc	Time(s)	Acc	Time(s)	Acc	Time(s)	Acc	Time(s)	Acc	Time(s)
SA	R6 Train	**0.958**	**200**	0.833	199	0.868	256	0.500	3075	0.848	1666	0.850	272
	R6 Test	**0.954**	**202**	0.766	209	0.854	251	0.520	3076	0.798	1736	0.842	285
NER	R7 Train	**0.917**	**580**	0.738	655	0.889	714	-	-	0.700	3675	0.775	785
	R7 Test	**0.916**	**599**	0.686	640	0.897	718	-	-	0.638	3720	0.692	789
QA	R8 Train	**0.975**	**530**	0.808	506	0.892	615	-	-	0.767	4281	0.825	669
	R8 Test	**0.905**	**532**	0.695	563	0.783	613	-	-	0.667	4310	0.726	702



(a) Continuous Optimization with No Constraint (b) ASCC (c) UAT (d) Ours

Figure 4: Loss Comparison on Trojan/Benign model

et al., 2021a) proposes to use syntactic structure as trigger to achieve stealthiness. *Combination Lock Attack* (Qi et al., 2021b) rephrases sentences by substituting a set of words with their synonyms and considers the substitution behaviors as the trigger. *SOS Attack* (Yang et al., 2021b) constructs a poisonous dataset by data augmentation to ensure the trigger sequence is only effective when all tokens appear in the input sentence. For each attack, we use 20 trojaned and 20 benign models. Specifically, hidden killer and combination lock models are trained on the SST-2 dataset with shared benign models (Socher et al., 2013), and the SOS models are trained on the IMDB movie review dataset (Maas et al., 2011a). All codes are from original Github repos.

Defense Setting. For backdoor detection, we use 20 samples in scanning, meaning that the trigger ought to flip majority of these samples. We invert a fixed number of tokens for all the attacks (10 in this paper). For the attacks which use longer triggers, we found that a subset of the ground-truth trigger tokens can already have a high Attack Success Rate, exposing the backdoors. Similar phenomenon is also reported in the literature (Yang et al., 2021b). For TrojAI models, we assume that defenders are not aware of the attack settings beforehand. Therefore, we need to enumerate all possible settings while launching our inversion method. For example, in TrojAI round7, there are 9 trigger types based on its positions, target label, etc. We try all possible combinations and consider a model trojaned if any combination yields a high ASR. We apply the same rule when evaluating all baselines. The time cost reported in the main text Table 1 is the aggregated time (of all combinations). For backdoor removal, we randomly select 20 trojaned models from the TrojAI training set in each round and first run our method to invert triggers. For the three advanced NLP attacks, we train 20 trojaned models per attack with different random seeds. We optimize the Eq.6 to unlearn the backdoors. Please see more details in Appendix A.

Baselines. We compare our method with 5 baselines for the backdoor detection task: Adversarial Sparse Convex Combination (ASCC) (Dong et al., 2021), Genetic Algorithm (GA) (Alzantot et al., 2018), Universal Adversarial Trigger (UAT) (Wallace et al., 2019), T-Miner (Azizi et al., 2021) and Adversarial Semantic Collisions (ASC) (Song et al., 2020). ASCC (Dong et al., 2021) is the most relevant technique to ours. It was proposed to generate NLP adversarial examples for adversarial training. For each word in the sentence, ASCC defines a convex hull over a small synonym list (less than 20) and uses gradient-based optimization to derive the word change. It also introduces a regularization item in the optimization objective to encourage the solution to be sparse. ASC (Song et al., 2020) is a gradient based method for generating Semantic Collisions:texts that are semantically unrelated but judged as similar by NLP models. It relaxes the discrete input space to a continuous one through *Softmax* function, then it leverages beam search to pick the token sequence. Similarly, UAT (Wallace et al., 2019) is also a gradient based inversion technique designed for generating backdoor triggers. To avoid the non-differentiable pipeline issue, UAT is applied on the embedding space. For each step, it generates an embedding vector for each token through back-propagation, then it projects the embedding to the most similar word embedding defined in the dictionary. GA (Alzantot et al., 2018) is a heuristic method, by defining the population (all possible token sequence with a fixed length), the instance and the score function (Attack Success Rate). GA tends to remain the instance with higher ASR for the next generation. T-Miner (Azizi et al., 2021)

is a model based trigger inversion technique. It leverages a sequence-to-sequence model to produce misclassified samples, then it analyzes the text produced by the generative model to determine if they contain trigger phrases. Their settings can be found in Appendix A.

5.2. Main Results

Backdoor Detection. We show the results on the 1548 TrojAI models in Table 1. The first and second columns indicate the tasks and the corresponding evaluation set. In columns 3-14, every two columns show the detection accuracy and the time cost for a method. As shown in the third and fourth columns, our method is able to achieve over 90% detection accuracy cross all tasks and outperforms all the baselines. For example, in the SA task, our method can achieve 0.958 and 0.954 on round 6 training and test sets, respectively. In contrast, the other three optimization based methods, ASCC, UAT and ASC, can only achieve $0.833/0.766/$, $0.868/0.854$ and $0.850/0.842$, respectively. T-Miner achieves 0.5 and 0.52 on the round 6 training and test sets. Note that T-miner is a generator based detection method and may not scale well to large transformer models. GA can get 0.848 and 0.798 accuracy on round 6 which are comparable to ASCC. For the round 7 NER task, we find that UAT is more effective than other baselines, it can achieve 0.897 on the test set, which is 21.1% better than ASCC, 20.5% better than ASC and 25.9% better than GA. However, our method is still 1.9% better. For the QA task, which is most challenging, our method has 0.905 accuracy on the test set, which is 21%, 12.2% 17.9% and 23.8% better than ASCC, UAT, ASC and GA, respectively. The time cost of our method is similar to the other three optimization methods due to the same setting, UAT is slightly slower than the other ours due to its projection operation in each step. The efficiency degradation for ASC is due to the beam search after optimization. T-Miner and GA are more costly. In particular, T-Miner takes $14\times$ more computation time for the round 6 SA models when compared to our method. It is because T-Miner fine-tunes a generator model when scanning each subject model. Our method is $8.6\times$, $6.21\times$ and $8.1\times$ faster than GA in the SA, NER and QA tasks.

Fig. 3 and Fig. 4 further explain the effectiveness of our method compared to the baselines. Fig. 3 illustrates the loss value changes over the optimization epochs for different methods on a trojaned QA model from TrojAI round 8. The green line is for ASCC. As illustrated in the methodology section, it quickly falls into a large false zone (FZ), then the entropy regularization item forces the optimizer to find a one-hot value in the FZ. The value yields a large loss since it is not the ground truth (GT). The blue line is for UAT, it leverages first-order Taylor-approximation to derive the gradient direction of the current optimization variable and directly projects it to the discrete input space. In other words, every point in the curve represents the loss of a one-hot token sequence. However, due to the rugged loss landscape, such local approximation might yield a wrong gradient direction, leading to the fluctuating curve and sub-optimal results. The red line is for our method. In the early epochs, it also falls in some FZ as ASCC does. Later, due to backtracking and randomization, it escapes the FZ's, reaches an optimal zone (OZ), and finally inverts the GT. Fig. 4 further compares the loss values between a trojaned and a benign SA models from TrojAI round 6 for each method. Due to the existence of FZ's, directly optimizing in the continuous space (with the default temperature) yields extremely small loss values for both trojaned and benign models as shown in Fig. 4(a), leading to a large number of false positives in scanning. Fig. 4(b) and 4(c) illustrate the differences for ASCC and UAT, respectively. Observe that due to the ineffective search of the GT in these methods, the trojaned model's loss is not distinguishable from the benign model's, leading to false negatives. In contrast, our method is able to achieve a clear separation between the benign and trojaned models as in Fig. 4(d).

Table 2: Backdoor Removal Results on TrojAI Datasets

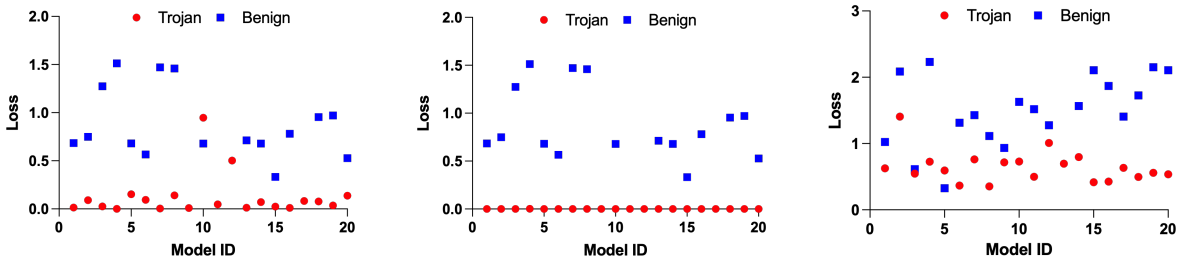
Task	Before Removal		After Removal	
	Clean Acc	ASR	Clean Acc	ASR
SA	0.909	0.980	**0.892**	**0.051**
NER	0.978	0.957	**0.959**	**0.032**
QA	0.727	0.998	**0.720**	**0.091**

Table 3: Backdoor Removal Results on Advanced attacks

Attack	Before Removal		After Removal	
	Clean Acc	ASR	Clean Acc	ASR
Hidden Killer	0.901	0.978	**0.899**	**0.095**
Combinational Lock	0.924	0.958	**0.926**	**0.072**
SOS	0.912	0.922	**0.907**	**0.256**

We evaluate our method on the 3 advanced NLP backdoor attacks. The results are in Fig. 5. In each subplot, we present the loss values of **optimal trigger estimations** by our method for individual models. The red cycles and blue boxes represent trojaned and benign models, respectively. As in Fig. 5(a) and Fig. 5(b), there are clear separations between trojaned and benign models for Hidden Killer and Combination Lock attacks. In particular, if we set $\beta = 0.3$, we are able to achieve 0.95 and 1.00 detection accuracy for the two attacks. The SOS attack can be considered an adaptive attack for our method. It substantially reduces the size of OZ via adversarial data augmentation, which makes it difficult for the optimizer to find the GT. As shown in Fig. 5(c), although the red and blue separation becomes much smaller than the other two attacks, we are still able to get 0.875 if we set the $\beta = 1$.

Backdoor Removal. We show the backdoor removal results on 60 randomly selected models for the 3 NLP tasks across



(a) Hidden Killer Attack (b) Combinational Lock Attack (c) SOS Attack

Figure 5: Evaluation on Advanced NLP Backdoor Attacks

Table 4: Adaptive Attack

Parameter	SA		NER		QA	
	Acc	ASR	Acc	ASR	Acc	ASR
$\phi = 0$	0.950	0.991	0.900	0.941	0.900	0.999
$\phi = 0.1$	0.900	0.959	0.900	0.942	0.900	0.962
$\phi = 0.5$	0.850	0.853	0.850	0.812	0.800	0.794
$\boldsymbol{\phi = 1}$	**0.650**	**0.531**	**0.550**	**0.448**	**0.600**	**0.428**

3 TrojAI rounds (20 for each round) in Table2. The first column presents the task. The second and third columns show the trojaned models' clean accuracy and attack success rate (ASR) on the test set before removal. The fourth and fifth columns show the average clean accuracy and ASR after removal. Observe that for all tasks, our method is able to reduce the ASR down to less than 10% with slight clean accuracy degradation. We further evaluate our removal method on the three advanced NLP backdoor attacks, using 20 trojaned models per attack. The results are shown in Table. 3. For all the attacks, our method can largely retain the clean accuracy. For Hidden Killer and Combinational Lock, our method can reduce the ASR to 0.072. For SOS, the most robust attack, our method can still reduce the ASR from 0.922 to 0.256.

Note that there are a number of highly effective backdoor removal methods in the visiom domain (Wang et al., 2019; Wu & Wang, 2021; Li et al., 2021b; Tao et al., 2022; Liu et al., 2018). These techniques cannot be easily adapted to the NLP domain which has different characteristics, such as sparse space, discrete pipeline, and sequential input.

5.2.1. Ablation Study

We study the contributions of our design choices on a subset of TrojAI models, with 20 trojaned and 20 benign for each task. The results show that the optimization has low precision scores without temperature scaling/reduction and low recall scores without backtracking. It renders the importance of both design choices. Please see details in Appendix B.

5.2.2. Adaptive Attack

Besides the SOS attack, we devise another adaptive attack which is targeting our Assumption II. The idea is to encourage samples stamped with triggers not to have a extremely small loss. In particular, we revise the target loss in Eq.1 as follows.

$$\mathcal{LD}_p(\theta^*) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathcal{L}(f(x;\theta^*), y) + \mathop{\mathbb{E}}_{(x^*,y^*)\sim\mathcal{D}^*} \max(\mathcal{L}(f(x^*;\theta^*), y^*) - \phi, 0) \tag{12}$$

where ϕ is a hyper-parameter to control the trigger loss during poisoning. For different ϕ values, we train 10 trojaned models and mixed them with the same number of benign models for each task. According to Table 4, our method stays effective when $\phi = 0.5$, the detection accuracy is over 80% for all tasks. As ϕ further increases, the method becomes less effective. However, the ASR drops a lot as well, which means the attack is not effective anymore. Intuitively, the ASR is entangled with the training loss. Therefore, it's difficult for attackers to achieve both high ASR and moderate loss value to bypass our method.

6. Conclusions

We present a novel optimization method for NLP model backdoor defense. It reduces the trigger inversion problem to inverting weight value vectors for all tokens in the dictionary. It then dynamically scales the temperature in the *softmax* function to gradually narrow down the search space for the optimizer, helping it to produce a close-to one-hot inversion result with high ASR. The one-hot value hence corresponds to the ground truth trigger. It also utilizes temperature backtracking to escape local optimals. Our experiments show that the method is highly effective and produces better backdoor scanning and removal results than the baselines.

7. Acknowledgment

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors.

References

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., and Chang, K.-W. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

Azizi, A., Tahmid, I. A., Waheed, A., Mangaokar, N., Pu, J., Javed, M., Reddy, C. K., and Viswanath, B. T-miner: A generative approach to defend against trojan attacks on dnn-based text classification. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

Bjerva, J., Bhutani, N., Golahn, B., Tan, W.-C., and Augenstein, I. Subjqa: A dataset for subjectivity and review comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.

Chen, C. and Dai, J. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*, 452:253–262, 2021.

Chen, X., Liu, C., Li, B., Lu, K., and Song, D. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

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

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020. URL https://openreview.net/pdf?id=r1xMH1BtvB.

Dai, J., Chen, C., and Li, Y. A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7: 138872–138878, 2019.

deepset. Deepset roberta. https://huggingface.co/deepset/roberta-base-squad2/, 2020.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018a. URL http://arxiv.org/abs/1810.04805.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018b.

Dong, X., Luu, A. T., Ji, R., and Liu, H. Towards robustness against natural language word substitutions. *arXiv preprint arXiv:2107.13541*, 2021.

Fan, M., Si, Z., Xie, X., Liu, Y., and Liu, T. Text backdoor detection using an interpretable rnn abstract model. *IEEE Transactions on Information Forensics and Security*, 16: 4117–4132, 2021. doi: 10.1109/TIFS.2021.3103064.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N., Peters, M., Schmitz, M., and Zettlemoyer, L. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

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

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 57–60, New York City, USA, June 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N06-2015.

IARPA. Trojai competition. https://pages.nist.gov/trojai/, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kurita, K., Michel, P., and Neubig, G. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*, 2020.

Li, S., Liu, H., Dong, T., Zhao, B. Z. H., Xue, M., Zhu, H., and Lu, J. Hidden backdoors in human-centric language models. *arXiv preprint arXiv:2105.00164*, 2021a.

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

Liu, K., Dolan-Gavitt, B., and Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294. Springer, 2018.

Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. Trojaning attack on neural networks. 2017.

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

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*,

abs/1907.11692, 2019b. URL http://arxiv.org/abs/1907.11692.

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

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011a.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Melamud, O., Goldberger, J., and Dagan, I. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pp. 51–61, 2016.

Mendivil, F., Shonkwiler, R., and Spruill, M. Restarting search algorithms with applications to simulated annealing. *Advances in Applied Probability*, 33(1):242–259, 2001.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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)*, pp. 188–197, 2019.

Peters, M. E., Ammar, W., Bhagavatula, C., and Power, R. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

Pruthi, D., Dhingra, B., and Lipton, Z. C. Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*, 2019.

Qi, F., Chen, Y., Li, M., Yao, Y., Liu, Z., and Sun, M. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369*, 2020.

Qi, F., Li, M., Chen, Y., Zhang, Z., Liu, Z., Wang, Y., and Sun, M. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*, 2021a.

Qi, F., Yao, Y., Xu, S., Liu, Z., and Sun, M. Turn the combination lock: Learnable textual backdoor attacks via word substitution. *arXiv preprint arXiv:2106.06361*, 2021b.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, art. arXiv:1606.05250, 2016.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675*, 2020.

Sang, E. F. and De Meulder, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

Shen, G., Liu, Y., Tao, G., An, S., Xu, Q., Cheng, S., Ma, S., and Zhang, X. Backdoor scanning for deep neural networks through k-arm optimization. *arXiv preprint arXiv:2102.05123*, 2021a.

Shen, L., Ji, S., Zhang, X., Li, J., Chen, J., Shi, J., Fang, C., Yin, J., and Wang, T. Backdoor pre-trained models can transfer to all. *arXiv preprint arXiv:2111.00197*, 2021b.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Song, C., Rush, A. M., and Shmatikov, V. Adversarial semantic collisions. *arXiv preprint arXiv:2011.04743*, 2020.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*, 2020.

Tao, G., Liu, Y., Shen, G., Xu, Q., An, S., Zhang, Z., and Zhang, X. Model orthogonalization: Class distance hardening in neural networks for better security. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

Tjong Kim Sang, E. F. and De Meulder, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pp. 142–147, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119195. URL https://doi.org/10.3115/1119176.1119195.

Turian, J., Ratinov, L.-A., and Bengio, Y. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1040.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wallace, E., Feng, S., Kandpal, N., Gardner, M., and Singh, S. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

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

Wang, R., Zhang, G., Liu, S., Chen, P.-Y., Xiong, J., and Wang, M. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pp. 222–238. Springer, 2020.

Weischedel, R. and Brunstein, A. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112, 2005.

Wu, D. and Wang, Y. Adversarial neuron pruning purifies backdoored deep models. *Advances in Neural Information Processing Systems*, 34, 2021.

Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C. A., and Li, B. Detecting ai trojans using meta neural analysis. *arXiv preprint arXiv:1910.03137*, 2019.

Yang, W., Li, L., Zhang, Z., Ren, X., Sun, X., and He, B. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *arXiv preprint arXiv:2103.15543*, 2021a.

Yang, W., Lin, Y., Li, P., Zhou, J., and Sun, X. 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)*, pp. 5543–5557, 2021b.

Zhang, Z., Xiao, G., Li, Y., Lv, T., Qi, F., Liu, Z., Wang, Y., Jiang, X., and Sun, M. Red alarm for pre-trained models: Universal vulnerabilities by neuron-level backdoor attacks. *arXiv preprint arXiv:2101.06969*, 2021.

Appendix

A. Detailed Experimental Settings

Tasks and Evaluation Models. We evaluate our method on 3 popular NLP tasks: Sentiment Analysis (SA) (Socher et al., 2013), Name Entity Recognition (NER) (Sang & De Meulder, 2003; Melamud et al., 2016) and Question Answering (QA) (Rajpurkar et al., 2016). The SA models are trained on 7 different datasets from Amazon review (Ni et al., 2019) and IMDB (Maas et al., 2011b) to output binary predictions (i.e., positive and negative). For NER, we consider the 540 TrojAI round 7 models, in which 180 from the training set and 360 from the test set. The datasets used to train these NER models include CoNLL-2002 (Tjong Kim Sang & De Meulder, 2003) with 4 name entities, the BNN corpus (Weischedel & Brunstein, 2005) with 4 name entities and OntoNotes (Hovy et al., 2006) with 6 name entities. A model is supposed to identify all the valid entities in given samples. For the QA task, we evaluate the 120 and 360 models from the TrojAI round 8 training and test sets, respectively. The QA models are trained on 2 public datasets: SQUAD V2 (Rajpurkar et al., 2016) and SubjQA (Bjerva et al., 2020). Given a pair of question and context, a QA model is supposed to predict the position of answer in the context. In total, there are 7 transformer architectures: DistilBERT (Sanh et al., 2019), GPT-2 (Radford et al., 2019), BERT (Devlin et al., 2018a), RoBERTa (Liu et al., 2019b), MobileBERT (Sun et al., 2020), Deepset (deepset, 2020) and Google Electra (Clark et al., 2020).

Attack Setting. *Standard Data Poisoning* (SDP) (Gu et al., 2017) is used for generating trojaned models in the TrojAI datasets. In particular, 9 poisoning configurations are used for the different tasks. Specifically, based on the attack goal, there are two types of attacks: *universal attack* which flips samples from all classes to the target class and *label specific attack* which only flips samples from a certain victim class to the target class. Based on the trigger effect range, there are *global trigger*, which is effective at any position in the input sentence, *local trigger* which can only cause misclassification when inserted at a certain position (range). For example, in the SA task, a local trigger can be set as effective only when it is in the first half of an input sentence. In the QA task, a sample local trigger can trigger misclassification only if it is stamped in the question. A trigger can be a single character, word or a short phrase.

Defense Setting. For backdoor detection, we use 20 samples per class in scanning, meaning that the trigger ought to flip majority of these samples. For SA, we use an auxiliary benign model during the optimization to insure that the inverted words are neutral (to avoid false positives). The benign model is randomly drawn from the TrojAI model set. For the NER models, as the number of labels is relatively large, we apply a strategy similar to K-Arm (Shen et al., 2021a), in which we run 20 epochs for each victim-target label pair and select the most promising top two pairs based on the loss value for further optimization. The same strategy is applied for all the optimization related baselines.

For backdoor removal, we randomly select 20 trojaned models from the TrojAI training set in each round and first run our method to invert triggers, then optimize based on Eq.6 to unlearn the backdoors. We use 10% of the training data and 20% of the chosen training samples are stamped with our inverted trigger and marked with the correct labels. The clean accuracy and ASR before and after removal are evaluated on the whole test set.

Baselines. We compare our method with 4 baselines for the backdoor detection task: ASCC (Dong et al., 2021), Genetic Algorithm (GA) (Alzantot et al., 2018), UAT (Wallace et al., 2019), and T-Miner (Azizi et al., 2021). ASCC (Dong et al., 2021) and GA (Alzantot et al., 2018) were proposed to generate NLP adversarial examples. They can be easily adapted for backdoor trigger inversion. In particular, we change their search space from small synonym lists to the entire dictionary. For ASCC, we set the coefficient of the sparsity regularization term to 10 as suggested in the paper. For GA, we set the population size to 300 and the mutation rate to 0.5 and the number of generations to 10. For UAT, we set k=1 in the top-k token search. ASCC and UAT are evaluated on all the models in the TrojAI datasets. For the other two costly methods, we evaluate GA on 248 SA models, 120 NER models and 180 QA models with half trojaned and half benign, and T-Miner only on the 248 SA models from TrojAI round 6. Note that T-Miner was proposed for the SA task and requires substantial efforts to extend it to other tasks.

Parameters Setting. We set the length of trigger to 10 (i.e., inverting 10 weight vectors). We set the number of optimization epochs to 200 and use the Adam (Kingma & Ba, 2014) optimizer with the initial learning rate 0.5. All optimization related baseline methods share the same configuration. All experiments are done on a machine with a single 24GB memory NVIDIA Quadro RTX 6000 GPU.

B. Ablation Study

We study the contributions of our design choices on a subset of TrojAI models, with 20 trojaned and 20 benign for each task. As shown in Table 5, without temperature scaling/reduction, the optimization keeps searching in the continuous space and converges inside false zones, leading to the low precision scores and hence the low accuracy. Without backtracking, the algorithm continues to reduce the temperature and eventually makes the optimizer walk in a very rugged (close to discrete) space. As a result, it produces many false

Table 5: Ablation Study

Method	QA			NER			QA		
	Precision	Recall	Acc	Precision	Recall	Acc	Precision	Recall	Acc
w/o Temp Scaling	0.559	0.950	0.600	0.581	0.900	0.625	0.563	0.900	0.600
w/o Backtracking	1.000	0.800	0.900	0.936	0.750	0.850	1.000	0.700	0.850
Ours	**1.000**	**0.950**	**0.975**	**0.947**	**0.900**	**0.925**	**1.000**	**0.850**	**0.925**

negatives, leading to the low recall scores and hence the low accuracy.