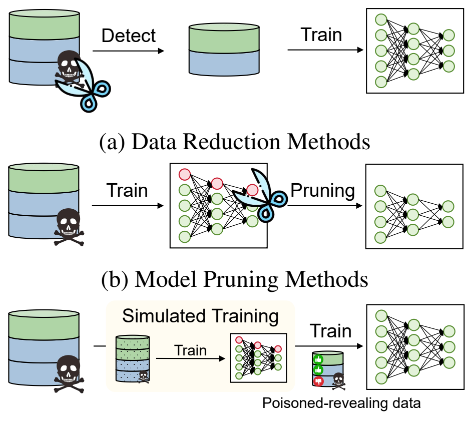
BadWindtunnel: Defending Backdoor in High-noise Simulated Training with Confidence Variance

Ruyi Zhang, Sonlei Jian*, Yusong Tan*, Heng Gao, Haifang Zhou, Kai Lu
National University of Defense Technology / Changsha, Hunan, China
{zhangruyi, jiansonglei, ystan, gaoheng21, haifang_zhou, kailu}@nudt.edu.cn

Abstract

Current backdoor attack defenders in Natural Language Processing (NLP) typically involve data reduction or model pruning, risking losing crucial information. To address this challenge, we introduce a novel backdoor defender, i.e., `BadWindtunnel`, in which we build a high-noise simulated training environment, similar to the wind tunnel, which allows precise control over training conditions to model the backdoor learning behavior without affecting the final model. We also use the confidence variance as a learning behavior quantification metric in the simulated training, which is based on the characteristics of backdoor-poisoned data (shorted in poisoned data): higher learnability and robustness. In addition, we propose a two-step strategy to further model poisoned data, including target label identification and poisoned data revealing. Extensive experiments demonstrate `BadWindtunnel`'s superiority, with a 21% higher average reduction in attack success rate than the second-best defender. The source code is public available at https://github.com/bettyzry/LBD.



(a) Data Reduction Methods

(b) Model Pruning Methods

(c) Our Method

Figure 1: The diagram of current methods versus ours: (a) data reduction methods, reducing poisoned information from the data; (b) model pruning methods, pruning poisoned neurons from the model; In contrast, (c) our method, revealing poisoned data through controlled simulated training and conducting defensive real training by fully utilizing both clean and poisoned information.

1 Introduction

Backdoor attacks are a significant security risk in NLP. These attacks manipulate a victim model that has good performance on clean data but always predicts the *target label* on *poisoned data*, which is applied with a specific *trigger* pattern. Triggers in NLP are categorized into four types: word, sentence, style, and syntax. Word and sentence triggers often involve inserting rare words or short sentences into the original text (Kurita et al., 2020; Chen et al., 2021; Dai et al., 2019). Style and syntactic triggers modify the text to match predefined styles or syntax (Qi et al., 2021b,c). These triggers have more implicit characteristics, allowing for a natural backdoor injection while preserving semantics. This semantic invariance makes such attacks particularly challenging to defend against.

As shown in Figure 1, current research mitigates backdoor attacks by reducing backdoor-related information from the data or model perspectives. As shown in Figure 1a, data reduction methods identify the statistical traits of poisoned data, such as spurious correlations with the target label, to eliminate poisoned data or remove embedded triggers (Yang et al., 2021; Qi et al., 2021a; He et al., 2023). However, these methods do not fully exploit the eliminated data and struggle with removing semantic-invariant attacks. In addition, residual poisoned data can still lead to a high attack success rate (Carlini et al., 2024), making this approach insufficient for an effective defense. As shown in Figure 1b, model pruning methods learn the differences in neuron activation between clean and poisoned data, enabling model pruning (Zheng et al., 2021; Tang et al., 2023; Yi et al., 2024). However,

*Corresponding author.

neurons often carry mixed information from both clean and poisoned data. Pruning may inadvertently eliminate some clean information, leading to a potential decline in overall model performance.

To overcome the above limitations, we propose an effective backdoor defense method, i.e., **Ba**ckdoor **d**efender in **Wind** **tunnel** (dubbed `BadWindtunnel`), that avoids data and model reductions, as shown in Figure 1c. In our method, we create a simulated training scheme, similar to a wind tunnel, to reveal poisoned data for real training. The simulated training does not affect the final model and allows precise control over training conditions to better model the learning behavior of poisoned data. The learning behavior during simulated training can be used to reveal poisoned data, guiding the gradient descent direction and rate in real training. Thus, `BadWindtunnel` manages backdoor attacks without reduction to data or pruning of the model, preserving all detailed information.

Specifically, in `BadWindtunnel`, we propose to use confidence variance instead of normal-used loss to quantify learning behavior in simulated training. The confidence variance indicates shifts in confidence throughout a training epoch, representing the model's learning speed, with poisoned data exhibiting higher values due to its greater learnability (Du et al., 2020; Hong et al., 2020; Li et al., 2021a). In addition, we control the noise rate in the simulated training to amplify the difference in confidence variance between the clean and poisoned data. Given the high robustness of poisoned data (Yang et al., 2021; Wei et al., 2024; Wu and Wang, 2021; Zhao et al., 2024), its learnability remains stable at the same noise level, making the confidence variance more distinguishable. To further reveal poisoned data, we propose a two-step strategy, including target label identification and poisoned data revealing. The strategy ensures that the confidence variance distribution is unaffected by semantic category, focusing solely on whether the data is poisoned. The main contributions are as follows:

- We propose an effective backdoor defender, i.e., `BadWindtunnel`, without reducing data or pruning neuron connections. It models poisoned data's learning behavior to generate poison-revealing data, guiding the gradient descent direction and rate in real training.

- We propose a novel backdoor defense scheme, i.e., simulated training, which isolates and controls the training environment to thoroughly explore poisoned data characteristics without affecting real training.

- We propose an easy but effective learning behavior quantification method, i.e., confidence variance, which leverages the high learnability of poisoned data and applies to multiple backdoor attacks.

Extensive experiments show that: (1) Our method effectively defends against the backdoor, achieving an average 21% lower attack success rate than the second-best defender and maintaining stable clean accuracy. (2) A comprehensive ablation study validates and visualizes the effectiveness of the simulated training, confidence variance, two-step strategy, and other key designs. (3) By increasing the proportion of poisoned data, we validate the robustness of the proposed model, which does not require any hyperparameter tuning.

2 Related Work

2.1 Characteristics of Poisoned Data

High Learnability. Many studies find that poisoned data are easier to learn because they only need to identify the triggers, unlike text semantics, which require full-text analysis. Zheng et al. (2021) observe that poisoned data often activate shortcuts from input to output layers. Other research shows that the average loss of backdoor samples drops faster than that of clean data after each training epoch (?Li et al., 2021a; Tang et al., 2023). In this paper, we also find that the confidence of poisoned data increases faster, further supporting this view.

High Robustness. Backdoor's high robustness is a key trick in its defenses. Due to the strong correlation between trigger and target label, backdoors are less affected by noise. Studies show that noisy poisoned data can still lead to stable outputs (Gao et al., 2019; Yang et al., 2021; Zhai et al., 2023), and noisy backdoored models also maintain consistent outputs for poisoned data (Wei et al., 2024). Even after random label scrambling and retraining, backdoor models retain the trigger-target label association (Cao et al., 2024). In our work, we leverage this robustness by creating a high-noise simulated training environment to amplify the learnability differences, enhancing defense effectiveness.

2.2 Methods for Backdoor Defense

Data Reduction Methods. Some researchers focus on modeling statistical features of poisoned data to eliminate poisoned data or triggers. RAP (Yang et al., 2021) and BDMMT (Wei et al., 2024) identify and remove data with high robustness. Xi et al. (2024) uses KL divergence in active space to distinguish poisoned data. ONION (Qi et al., 2021a) removes words that increase text perplexity. He et al. (2023) eliminate words falsely associated with the target label. Qi et al. (2021c) removes triggers through multiple translations. However, these methods struggle with semantic-invariant attacks and waste the information in poisoned data.

Model Pruning Methods. Liu et al. (2018) try to prune neurons inactive during clean data training. Li et al. (2021b) tries to reconstruct the compromised neural network through knowledge distillation. Wu and Wang (2021) employ adversarial weight perturbation to enhance the difference between clean and malicious neurons. Tang et al. (2023) use the model's shallow network to create a honeypot for backdoor attacks. MuScleLoRA encourages models to prioritize learning high-frequency clean mappings (Wu et al., 2024). However, these pruning methods necessitate a comprehensive understanding of the attacked model, limiting their generalizability and usability. Moreover, pruning may eliminate some correct connections, potentially reducing model performance.

In conclusion, current methods often use removal or pruning for defense, leading to information loss. This paper proposes a non-reductive training scheme to leverage the high learnability and robustness, enhancing defense effectiveness.

3 Method

3.1 Problem Formulation

Backdoor Attacks. Given a raw dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=0}^{N}$ with N samples, where x_i is a text and y_i is the semantic label. We set $\boldsymbol{y} = \{y_0, y_i, \dots, y_N\}$. Backdoor attack aims to manipulate the outputs of the victim model M to satisfy:

$$M(x_{\text{input}}) = \begin{cases} y_i, & x_{\text{input}} = x_i, \\ y_{\text{target}}, & x_{\text{input}} = \tau(x_i), \end{cases} \quad (1)$$

in which $\tau(\cdot)$ denotes the trigger injection function, and y_{target} is a predefined target label.

The typical backdoor attack usually employs data poisoning. Specifically, given an attack rate $r = \frac{N'}{N}$, the attacker randomly selects a subset $\mathcal{I}$ from the dataset $\mathcal{D}$ and poisons it into $\mathcal{I}' = \{(\tau(x_i), y_{\text{target}})\}_{i=0}^{N'}$. The poisoned dataset, $\mathcal{D}_{\text{poison}} = (\mathcal{D} - \mathcal{I}) \cup \mathcal{I}'$, is then used to train M. Thus, data in $\mathcal{D}_{\text{poison}}$ also contains a hidden label $y_{\text{poison},i}$, which indicates the poisoned status with 1 denotes poisoned. Notably, such attacks are often implicit, with the attacker choosing data with semantic label $y_i = y_{\text{target}}$ rather than forcibly changing it. This implicit poisoning aligns all texts and labels in $\mathcal{D}_{\text{poison}}$, increasing defense complexity.

Backdoor Defends. Upon receiving $\mathcal{D}_{\text{poison}}$, the defenders do not know which data are attacked, nor do they have any knowledge of the target label y_{target} or specific trigger patterns $\tau(\cdot)$. The ultimate goal of the defenders is to identify and cleanse the poisoned inputs $\tau(x_i)$ or reform the victim model M, enabling it to predict the real semantic labels y_i without affecting the performance on clean data. Appendix A summarizes main notations.

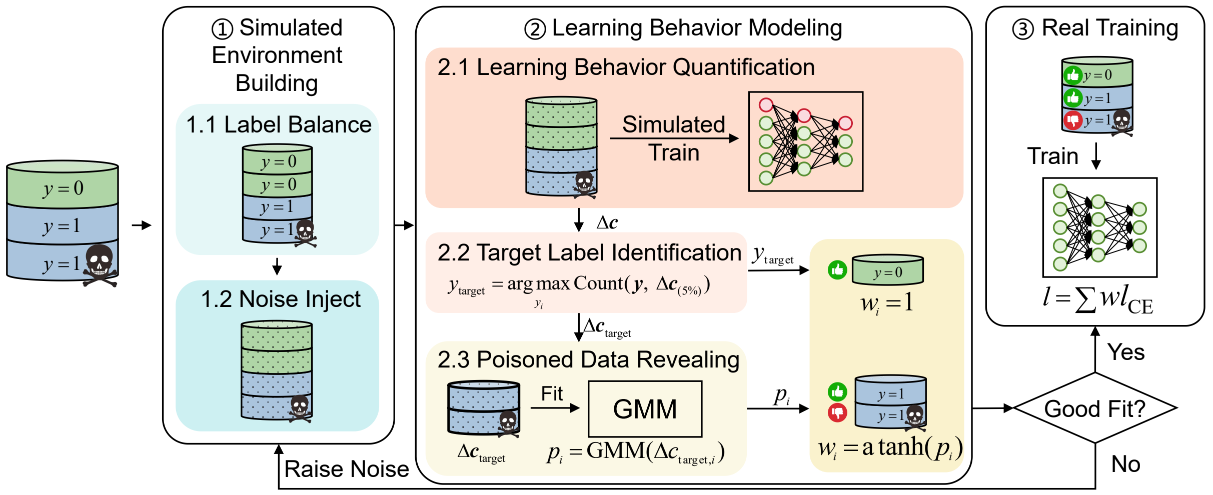
3.2 Overview

Figure 2 illustrates the framework of our proposed BadWindtunnel, which consists of three main steps: (1) **Simulated Environment Building**: We construct the environment by label balancing and noise injection to build $\mathcal{D}'_{\text{poison}}$ with amplified distinction in learning behaviors. (2) **Learning Behavior Modeling**: We generate the confidence variance by simulate training on $\mathcal{D}'_{\text{poison}}$ to quantify the learning behavior. Then, use it to identify the target label. We fit target data with a Gaussian mixture model (GMM), using GMM's posterior probability to generate poison-revealing data $\mathcal{D}^*_{\text{poison}}$. The modeling quality is evaluated. If the modeling is substandard, revert to the environment building step and increase the noise rate. Otherwise, proceed to (3) **Real Train**: Use $\mathcal{D}_{\text{poison}}$ to produce a clean model by loss-corrected real training.

3.3 Simulated Environment Building

We first establish a simulated environment with controlled training conditions. It transforms a raw poisoned dataset, $\mathcal{D}_{\text{poison}}$, into $\mathcal{D}'_{\text{poison}}$ to augment both the modeling accuracy and defense efficacy in subsequent learning behavior modeling. This process consists of two stages:

Label Balancing. The data quantity across categories may be imbalanced in the raw data, leading to uneven training frequency and distinct confidence variances between categories. To mitigate



① Simulated Environment Building
1.1 Label Balance
1.2 Noise Inject
Raise Noise
② Learning Behavior Modeling
2.1 Learning Behavior Quantification
Simulated Train
Δc
2.2 Target Label Identification
y_target = arg max Count(y, Δc_(5%))
Δc_target
2.3 Poisoned Data Revealing
Fit
GMM
p_i = GMM(Δc_target,i)
p_i
w_i = 1
w_i = a tanh(p_i)
③ Real Training
Train
l = Σ w l_CE
Good Fit?
Yes
No
y = 0
y = 1

Figure 2: The framework of BadWindtunnel. Take $y_{\text{target}} = 1$ for example. ① Given poisoned data, BadWindtunnel initially builds the simulated training environment through label balancing and noise injection. ② The processed data undergoes simulated training, quantifying learning behavior. Subsequently, the poisoned data is revealed and assigned weights through a two-step strategy: target label identification and poisoned data revealing. ③ Finally, we use the poison-revealing data for loss-corrected real training.

this, we initially balance the labels by upsampling underrepresented categories.

Noise Injection. As the poisoned data have high robustness, the text semantics are susceptible to equivalent noise rate. By introducing artificial noise, we can accentuate the learnability difference between clean and poisoned data, enhancing the accuracy of subsequent modeling. We achieve this by randomly swapping the order of letters in $t\%$ of the words within the text. The noise rate increases with the number of simulated training. Details are shown in Appendix B.

3.4 Learning Behavior Modeling

In this section, we first generate confidence variance by simulated training on $\mathcal{D}'_{\text{poison}}$ to quantify learning behaviors. We then utilize this confidence variance in two phases: target label identification and poisoned data revealing, resulting in poison-revealing data $\mathcal{D}^*_{\text{poison}} = \{(x_i, y_i, w_i)\}_{i=1}^{N}$.

3.4.1 Learning Behavior Quantification

As shown in Figure 3, the confidence variance $\Delta \boldsymbol{c}$ of poisoned data is significantly larger than that of the clean data. Thus, we use it to quantify learning behavior. Given a text x_i, its confidence c_i is:

$$c_i := \max(h'_0, h'_1, \ldots, h'_Y), \qquad h'_n = \text{Softmax}(h_n) = \frac{\exp(h_n)}{\sum_m^Y \exp(h_m)}, \tag{2}$$

in which $h_n = P(y_i = n|x_i)$ is the probability of the model classifying x_i into category n. Y is the amount of semantic categories. We can calculate the confidence variance Δc_i as:

$$\Delta c_i = c_i(1) - c_i(0), \tag{3}$$

where $c_i(0)$ and $c_i(1)$ are the confidence values before and after a training epoch, respectively.

Notably, many studies recognize the high learnability of poisoned data with loss variance (Li et al., 2021a; Tang et al., 2023). In contrast, we use confidence as the quantification standard. By definition, confidence reflects the model's assurance in classifying text into "any category", whereas loss indicates the certainty of the "real category". Confidence can reveal numerical differences more easily and earlier without requiring higher model capability or deeper learning. Thus, we prefer confidence variance for simplicity and early detection.

3.4.2 Target Label Identification

We identify the target label rather than model the raw data directly, as the raw data contains two distinct distributions: (1) semantic category and (2) poisoned status. Isolating the target label helps remove bias from the semantic category, ensuring a purer distribution for more accurate results.

Identifying the target label leverages the observation that the poisoned data have higher confidence variances. Define Count$(\boldsymbol{y}, C)$ as the number of category's occurrences under condition C. The target label y_{target} is the category with the highest proportion in the top 5% of data with the most significant confidence variance $\Delta\boldsymbol{c}_{(5\%)}$:

$$y_{\text{target}} = \arg\max_{y_i} \text{Count}(\boldsymbol{y}, \Delta\boldsymbol{c}_{(5\%)}). \quad (4)$$

This way, the non-target data can be immediately classified as clean with weights $w_i = 1$. We only need to model the target data's confidence variance $\Delta\boldsymbol{c}_{\text{target}} = \{\Delta c_i | y_i = y_{\text{target}}, y_i \in \boldsymbol{y}\}$.

3.4.3 Poisoned Data Revealing

As shown in Figure 3, the poisoned status significantly affects the confidence variance distribution. We aim to leverage this feature to reveal poisoned data easily, efficiently, and effectively without hyperparameters. Gaussian Mixture Models (GMMs), a popular unsupervised technique (Reynolds et al., 2009; Arazo et al., 2019), are used for this purpose. The probability density function of a GMM with K components in terms of confidence variance $\Delta\boldsymbol{c}_{\text{target}}$ is defined as:

$$P(\Delta\boldsymbol{c}_{\text{target}}) = \sum_{k=1}^{K} P(\Delta\boldsymbol{c}_{\text{target},k}), \quad (5)$$

in which $\Delta\boldsymbol{c}_{\text{target},k}$ is the set of $\Delta\boldsymbol{c}_{\text{target}}$ belonging to class k and $\Delta\boldsymbol{c}_{\text{target},k} \sim \mathcal{N}(\mu_k, \sigma_k^2)$. We use a two-component (clean-poisoned) GMM to model the distribution. The probability p_i of x_i being poisoned, calculated by its corresponding $\Delta c_{\text{target},i}$, is determined using GMM's posterior probability:

$$\begin{aligned} p_i &= P(k = 1 | \Delta c_{\text{target},i}) \\ &= \frac{P(\Delta c_{\text{target},i} | k = 1) P(k = 1)}{P(\Delta c_{\text{target},i})}, \end{aligned} \quad (6)$$

where $k = 0$ (1) denotes a clean (poisoned) data.

We aim to adjust the loss weight of each x_i in real training based on p_i. The weight should be negative if the data is likely poisoned ($p_i \to 1$), positive if it is likely safe ($p_i \to 0$), and have minimal impact if its poisoned probability is unclear ($p_i \to 0.5$). The soft weight can be achieved using the inverse hyperbolic tangent function:

$$w_i = \text{atanh}(p'_i) = \frac{1}{2} \ln\left(\frac{1 + p'_i}{1 - p'_i}\right), \quad (7)$$

where $p'_i = -1.6 p_i + 0.8$ is a linear transformation of p_i. Since the function is monotonically increasing in $[-1, 1]$ and $p_i \in [0, 1]$, we use this scaling to ensure the maximum weight is 1, maintaining consistency with the weights of non-target data.

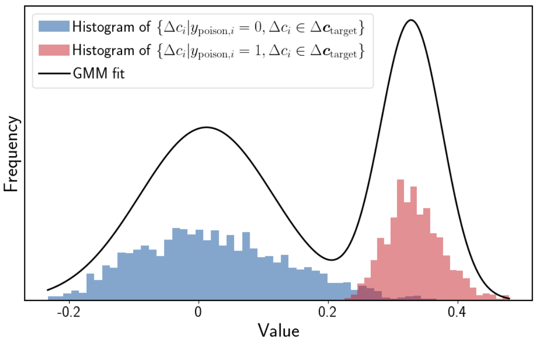


Figure 3: The actual distribution and GMM fitting results of confidence variance for clean and poisoned data within the target data. The poisoned status is indicated by y_{poison}, with 0 for clean and 1 for poisoned, and are colored for clarity.

By combining the weights from Section 3.4.2 and Section 3.4.3, we get the poison-revealing data $\mathcal{D}^*_{\text{poison}} = \{(x_i, y_i, w_i)\}_{i=0}^{N}$.

3.5 Real Train

We design an iterative mechanism to improve defense efficacy. If the modeling is unsatisfactory, we introduce additional noise and return to the simulated environment building phase; otherwise, it proceeds to the real training phase. The modeling quality is quantified by the Davies-Bouldin Index (DBI), with a detailed explanation in Appendix C.

The real training aims to maximize information utilization from all data. It involves gradient descent on clean data and gradient ascent on poisoned data. The corrected loss based on $\mathcal{D}^*_{\text{poison}}$ is:

$$l = \sum_{(x_i, y_i, w_i) \in \mathcal{D}^*_{\text{poison}}} w_i L_{\text{CE}}(M_\theta(x_i), y_i). \quad (8)$$

This approach avoids any data reduction or model pruning. Comprehensively using the information enhances the defense efficacy of our method.

The time complexity of `BadWindtunnel` is analyzed in Appendix D.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on four widely used text classification datasets: **SST-2** (Socher et al., 2013), **HSOL** (Davidson et al., 2017), **IMDB** (Maas et al., 2011) and **AG News** (Zhang et al., 2015). The former three are binary-classification datasets, while AGNews is a four-classification dataset. More details can be found in Appendix E.

Data	Attack	No-Defense		ONION		RAP		Z-SEQ		MuScleLoRA		BadActs		Ours	
		CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓
SST-2	BadNets	91.31	96.27	87.29	19.34	90.21	73.93	**91.24**	6.84	83.25	21.62	88.90	0.68	90.21	**0.00***
	AddSent	91.22	100.00	87.46	93.55	90.17	100.00	91.15	20.57	81.65	45.59	89.25	37.04	**91.37**	**0.00***
	Stylebkd	90.06	85.20	84.16	85.71	89.12	84.80	87.91	70.55	82.42	37.11	88.34	70.94	**89.21**	**32.87**
	Synbkd	90.09	95.42	84.96	95.39	79.21	90.13	77.57	33.18	84.22	28.73	88.15	87.21	**88.81**	**7.18**
HSOL	BadNets	89.52	99.01	89.26	8.57	71.60	59.35	89.52	30.15	89.54	1.15	89.54	0.30	**91.57**	**0.00***
	AddSent	91.59	99.99	91.16	94.18	91.29	99.99	**91.55**	5.23	89.54	11.93	89.09	16.94	91.48	**0.00***
	Stylebkd	89.08	85.47	87.68	80.76	88.76	84.91	76.58	86.59	**89.42**	**18.19**	86.37	62.77	89.05	31.91
	Synbkd	90.28	98.27	89.53	90.18	89.52	98.12	83.67	86.11	89.34	**0.80**	88.52	39.83	**90.75**	5.72
IMDB	BadNets	93.93	86.44	**93.55**	49.26	92.36	65.94	86.59	6.37	86.43	13.08	92.07	12.03	92.52	**1.19**
	AddSent	93.90	96.33	**93.55**	66.25	83.93	77.50	84.99	**7.09**	86.33	14.32	92.11	67.31	92.55	9.47
	Stylebkd	93.74	98.93	**93.32**	99.20	91.76	98.64	50.06	81.05	86.11	64.40	91.91	0.06	91.32	**0.03**
	Synbkd	93.73	41.67	**93.37**	42.21	83.29	38.43	89.53	26.78	86.25	33.47	91.89	**3.28**	91.67	3.82
AGNews	BadNets	93.97	83.55	92.69	6.75	**93.70**	59.67	92.16	0.28	88.35	2.03	92.12	92.83	88.39	**0.00***
	AddSent	94.32	100.00	93.16	81.62	79.94	80.00	83.29	0.01	87.97	99.96	92.19	72.95	**93.44**	**0.00***
	Stylebkd	94.32	100.00	**93.16**	81.62	79.94	80.00	83.29	**0.01**	87.97	99.96	92.19	72.95	92.26	32.52
	Synbkd	94.24	99.83	93.23	96.52	**93.61**	99.79	85.22	**5.87**	87.46	97.29	92.20	57.24	93.38	6.91
Average		92.21	91.65	90.47	68.19	86.78	80.70	84.02	29.17	86.64	36.85	90.30	43.40	**91.12**	**8.23**

Table 1: CACC and ASR on four datasets with four attackers. Six defenders are tested. The grayed out records the raw model without defense. All the results are in %. The best are in **bold** and the second best are in underline. $\star$ denotes the attainment of the theoretical optimum outcome.

Backdoor Attackers. We evaluate our method with four widely used attackers: (1) **BadNets** selects rare word like "*cf*" as a trigger and randomly inserts it into the text (Kurita et al., 2020). (2) **AddSent** uses short sentence like "*I watch this 3D movie*" as a trigger and randomly inserts it into the text (Dai et al., 2019). (3) **Stylebkd** employs Bible style as a trigger (Qi et al., 2021b). (4) **Synbkd** uses predefined syntactic template as a trigger (Qi et al., 2021c). All attackers are implemented by the open-source project OpenBackdoor[1].

Backdoor Defenders. We compare our method against five backdoor defenders, each addressing different aspects like data reduction and model pruning. These include: (1) ONION minimizes the words contributing to text confusion (Qi et al., 2021a). (2) RAP reduces the texts with strong prediction robustness to noise (Yang et al., 2021). (3) Z-SEQ reduces words with spurious correlations to the target label (He et al., 2023). (4) MuScleLoRA encourages the model to prioritize the high-frequency clean mappings (Wu et al., 2024). (5) BadActs purifies the poisoned data by aligning abnormal activations with optimized clean activation intervals (Yi et al., 2024). All defenders are tested using their open-source codes and default hyperparameters.

[1]https://github.com/thunlp/OpenBackdoor

Evaluation Metrics. We evaluate defense performance using two metrics: (1) **Clean Accuracy (CACC)**: The likelihood of victim model correctly classifying clean data. A good defense should maintain a high CACC, close to the original undefended value. (2) **Attack Success Rate (ASR)**: The likelihood of the victim model misclassifying poisoned data as the target label. A successful defense should achieve a lower ASR.

Parameter Settings and Implementations. Our experiments are conducted on a workstation with an Intel Xeon Silver 6230R CPU, three NVIDIA A40 GPUs, and 503 GB of RAM. We employ the widely-used BERT-base-uncased model (Devlin et al., 2019). The attack rate is set at 20%, consistent with the original attack settings (Qi et al., 2021b; Yi et al., 2024). We use the poisoned data for five epochs of backdoor training with a learning rate of $2e-5$ to get the poisoned model.

4.2 Effectiveness Test

We test BadWindtunnel's defense effectiveness against backdoor attacks using four datasets, each exposed to four different attackers. No defense and five other defenders are compared. Each experiment runs five times independently, and we log the average CACC and ASR for reliability.

Table 1 displays the experimental results, with BadWindtunnel consistently excelling across in all

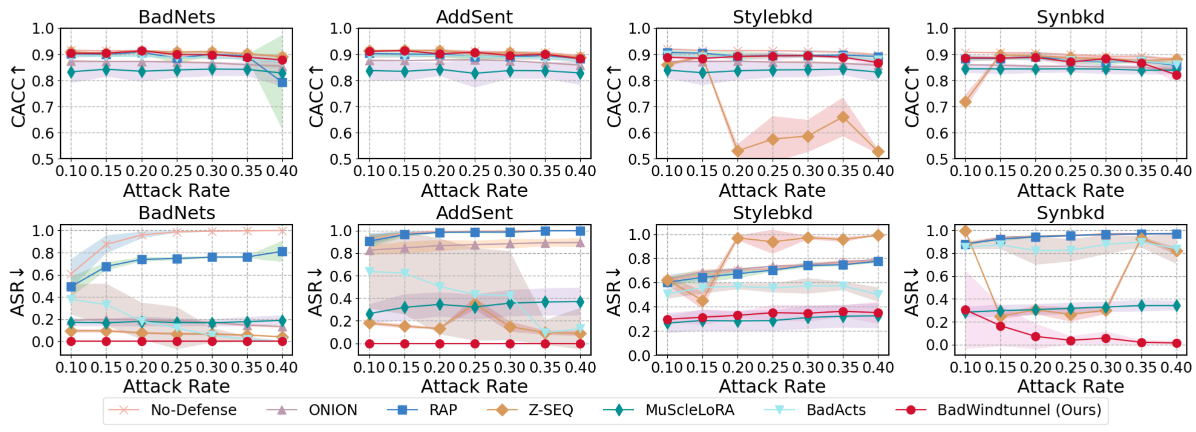


Figure 4: Robustness test. Defensive results of different defenders against backdoor attack rates of 10%-40% under four attackers. Each column represents an attacker, each row corresponds to an evaluation metric, and each line represents a defender. Mean and standard deviation values are plotted for each case.

settings and metrics. It achieves a top-two ASR reduction in all cases and reaches the theoretically optimal value in 6 cases, with an average ASR reduction that is 21% higher than that of the next-best defender. In maintaining CACC, it secures top-two results in 14 out of 16 cases, boasting the highest average CACC. Although `Z-SEQ` outperforms `BadWindtunnel` in some ASR reduction cases in AGNews, this comes with a significant CACC decrease. The low ASR on most defenders on IMDB dataset is due to its longer texts characteristics. Stylebkd and Synbkd modifying entire texts are less effective on long texts. Therefore, low initial ASR limits defense potential. Besides, most methods experience a notable performance decline against Stylebkd and Synbkd attacks, while `BadWindtunnel` maintains a stable defense. These results underscore `BadWindtunnel`'s superiority in backdoor defense, attributed to its full utilization of data and model information.

4.3 Robustness Test

We evaluate the robustness of `BadWindtunnel` and five competitors against attack rates ranging from 10% to 40%, which represent the fraction of the poisoned data in $\mathcal{D}_{\text{poison}}$. These evaluations use the SST-2 dataset, a standard NLP resource, ensuring a precise assessment of each method's robustness.

Figure 4 presents the average and standard deviations of the CACC and ASR from five independent runs. While most methods' defenses deteriorate as attack rates increase, our method consistently performs well. Notably, under the Synbkd attack, `BadWindtunnel`'s ASR is higher at lower attack rates, possibly because the smaller proportion of poisoned data allows clean data to catch up in learnability through additional training. Despite this, `BadWindtunnel`'s ASR still ranks second, emphasizing its stable defense. Interestingly, `BadActs`' performance improves as the attack rate increases, a phenomenon not discussed in its original paper. We verified this result using the provided source code, only altering the attack rate. Overall, these results highlight `BadWindtunnel`'s effective and stable defense against backdoor attacks.

Metrics	Defenders	Attackers			
		BadNets	AddSent	Stybkd	Synbkd
CACC ↑	`BadWindtunnel`	**91.6**	**91.5**	**89.1**	**90.8**
	`w/o` `Lab.Bal.`	-2.3	-1.9	-34.3	-46.2
	`w/o` `N.Inject`	-27.9	-22.3	-0.1	-38.1
	`w/o` `TwoStep`	-37.7	-41.8	-15.0	-25.7
	`w/o` `SoftWeight`	-2.0	-1.9	-0.2	-2.0
	`w/o` Δc	-16.9	-9.4	-4.6	-17.7
ASR ↓	`BadWindtunnel`	**0.0**	**0.0**	**31.9**	**5.7**
	`w/o` `Lab.Bal.`	+99.8	+100.0	+42.2	+92.7
	`w/o` `N.Inject`	+70.4	+41.2	+11.8	+74.6
	`w/o` `TwoStep`	+1.0	0.0	+32.6	+2.2
	`w/o` `SoftWeight`	+99.4	+99.9	+12.5	+82.9
	`w/o` Δc	+20.0	+40.0	+22.6	+35.7

Table 2: Ablation study. Five simplified versions on four attackers are tested. We record the CACC and ASR on the original `BadWindtunnel`, as well as changes in the corresponding values on the simplified versions. All the results are in %. The best are in **bold**.

4.4 Ablation Study

We assess the key designs of `BadWindtunnel` with four attack methods through five simplified versions: **`w/o`** `Lab.Bal.` (without label balance), **`w/o`** `N.Inject.` (without noise injection), **`w/o`** `TwoStep` (without the two-step strategy),

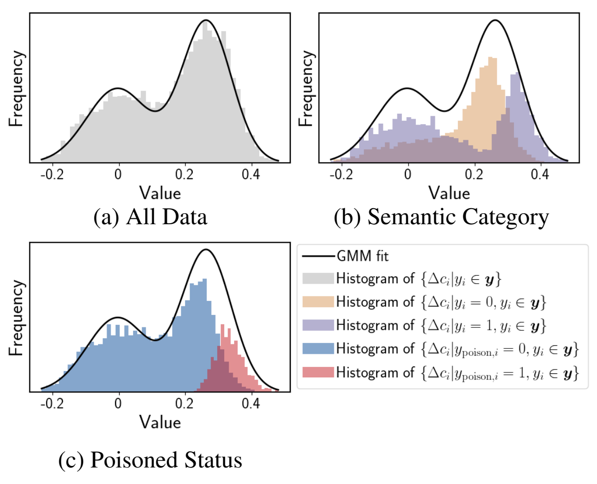


Figure 5: Confidence variance distribution in all data. y_i represents the semantic category. $y_{\text{poison},i}$ represents the poisoned status label, where 0 is clean and 1 is poison. The figure takes $y_{\text{target}} = 1$ as an example.

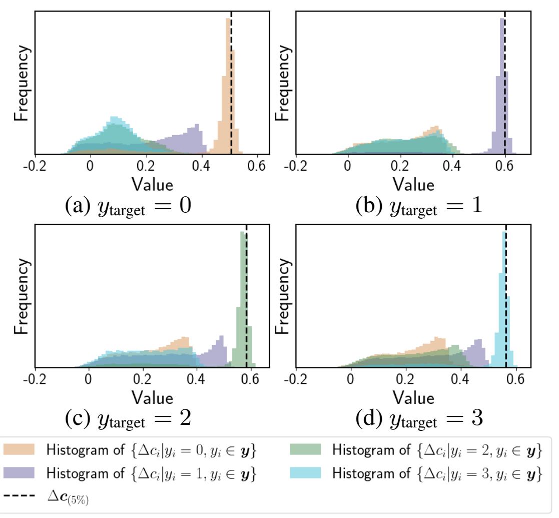


Figure 6: Visualization of target label identification. We present the distribution and the 5% percentile point of confidence variance. Four categories are color-coded for clarity. Each sub-graph represents a target label setting. y_i indicates the semantic label.

w/o SoftWeight (directly use poisoned probability as weights), and **w/o** Δc (substituting loss for confidence). We use HSOL for evaluation, chosen for its label imbalance, to highlight the effectiveness of label balancing. Each experiment runs five times independently, with average performance recorded.

Table 2 shows that BadWindtunnel notably surpasses all the simplified versions, validating its key designs. Omitting any of these features results in a considerable increase in ASR. Specifically, excluding label balancing and noise injection considerably reduces the CACC, as these elements enhance learning behavior. Removing the two-step strategy also leads to a sharp decrease in CACC, as the learning behavior, influenced by the semantic category, results in incorrect poisoned status identification. Replacing soft weights with hard weights causes a significant increase in ASR, with negligible CACC changes, suggesting that while hard weights align with the defense strategy, they can be overly absolute. Substituting loss for confidence has a minor effect but is less effective than the original method, indicating the superiority of the proposed confidence model. In summary, all the proposed key designs are vital for the defense's effectiveness.

4.5 Visualization Test

We visualize the importance of the two-step strategy, i.e., identifying the target label first and then revealing the poisoned data. Figure 5a shows the confidence variance Δc distribution and the fitted GMM curve for all datasets. The distribution can be divided into two types: (1) the semantic category (shown in Figure 5b) and (2) the poisoned status (shown in Figure 5c). Directly modeling the mixed distribution may overly focus on the semantic category, leading to incorrect GMM fitting and failing to distinguish the poisoned status. On the contrary, as shown in Figure 3, identifying the target label first allows GMM to focus on the poisoned status, accurately revealing the poisoned data.

We visualize the effectiveness of confidence variance in identifying the target label. Figure 6 shows the confidence variance distribution for different categories and the $\Delta c_{(5\%)}$ percentile point, with various target label sets. Due to space limitations, we only display the results on AGNews under BadNets attack. AGNews is chosen because it is a four-class dataset, making target label identification more challenging and validating our method robustly. More experiments are provided in Appendix F.1. In all scenarios, the category corresponding to the target label shows a significant increase in confidence variance and constitutes the majority within the 5% percentile, confirming the effectiveness of confidence variance.

The ability of confidence variance to reveal the poisoned status is visualized in Figure 3. Additional experiments on all datasets are provided in Appendix F.2.

5 Conclusion

This paper introduces BadWindtunnel, a novel defense scheme for backdoor attacks in NLP. It builds a high-noise simulated training environment to reveal poisoned data without data reduction or model pruning. Precisely, we quantify the high learnability of poisoned data using confidence variance and model it with the GMM in a two-step strategy. The poison-revealing data guides the gradient descent direction and rate during the defensive real training. Experimental results show that BadWindtunnel reduces the attack success rate by an additional 21% compared to the second-best defender on average and demonstrates robustness. Ablation and visualization experiments further validate the effectiveness of our key designs.

6 Limitations

We propose a confidence variance-based simulated training in the high-noise environment against backdoor attacks in NLP. Our approach uses the learnability of poisoned data to quantify learning behavior and reveals the poisoned data to guide the defensive real training.

However, the learnability of poisoned data poses a limitation when the clean data vastly outnumber it. Over more training chances, clean data can match the learning progress of the poisoned data, thus masking their learning behavior differences. We mitigate the quantity disparity from category imbalance via label balancing but cannot eliminate it when the attack rate is extremely low. As shown in Section 4.3, BadWindtunnel maintains optimal performance under most attacks when the attack rate is 10%. However, its performance starts to decrease for semantic invariant attacks like Synbkd.

Given this limitation, future research could explore further ways to magnify the differences in learnability. For instance, we can try to increase the learning difficulty of clean data by creating more complex artificial noise or investigating different perturbation methods. Alternatively, efforts could be made to enhance the learnability of the poisoned data, such as implementing controlled loss reduction in simulated training.

7 Ethics Statement

Our study introduces an efficient method to protect NLP models from backdoor attacks. We believe that our proposed method will contribute to mitigating security risks associated with such attacks. All experiments conducted in this paper utilize established open datasets. We do not anticipate any direct negative consequences to the work, and we hope to expand upon our research and advance the development of more robust defense methods in future investigations.

Acknowledgments

This work is sponsored in part by the NUDT foundation under Grant No.24-ZZCX-JDZ-07 and National Natural Science Foundation of China under Grant No. 62421002.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel O'Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, pages 312–321. PMLR.

Bochuan Cao, Yuanpu Cao, Lu Lin, and Jinghui Chen. 2024. Defending against alignment-breaking attacks via robustly aligned llm. *Annual Meeting of the Association for Computational Linguistics*.

Nicholas Carlini, Matthew Jagielski, Christopher A. Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning web-scale training datasets is practical. In *IEEE European Symposium on Security and Privacy*, pages 407–425. IEEE.

Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Annual Computer Security Applications Conference*, pages 554–569.

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

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *AAAI Conference on Artificial Intelligence*, volume 11, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Associationfor Computational Linguistics*.

Min Du, Ruoxi Jia, and Dawn Song. 2020. Robust anomaly detection and backdoor attack detection via differential privacy. *The International Conference on Learning Representations*.

Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. STRIP: A defence against trojan attacks on deep neural networks. In *Annual Computer Security Applications Conference*, pages 113–125.

Xuanli He, Qiongkai Xu, Jun Wang, Benjamin Rubinstein, and Trevor Cohn. 2023. Mitigating backdoor poisoning attacks through the lens of spurious correlation. In *Conference on Empirical Methods in Natural Language Processing*, pages 953–967.

Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. 2020. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv:2002.11497*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Annual Meeting of the Association for Computational Linguistics*.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021a. Anti-backdoor learning: Training clean models on poisoned data. *Conference on Neural Information Processing Systems*, 34:14900–14912.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021b. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *The International Conference on Learning Representations*.

Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics*, HLT '11, pages 142–150, USA.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021a. ONION: A simple and effective defense against textual backdoor attacks. In *Conference on Empirical Methods in Natural Language Processing*.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021b. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Conference on Empirical Methods in Natural Language Processing*.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021c. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In *Annual Meeting of the Association for Computational Linguistics*.

Douglas A. Reynolds et al. 2009. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663).

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Ruixiang Ryan Tang, Jiayi Yuan, Yiming Li, Zirui Liu, Rui Chen, and Xia Hu. 2023. Setting the trap: Capturing and defeating backdoors in pretrained language models through honeypots. *Conference on Neural Information Processing Systems*, 36:73191–73210.

Jiali Wei, Ming Fan, Wenjing Jiao, Wuxia Jin, and Ting Liu. 2024. BDMMT: Backdoor sample detection for language models through model mutation testing. *IEEE Transactions on Information Forensics and Security*, 19:4285–4300.

Dongxian Wu and Yisen Wang. 2021. Adversarial neuron pruning purifies backdoored deep models. In *Conference on Neural Information Processing Systems*, volume 34, pages 16913–16925.

Zongru Wu, Zhuosheng Zhang, Pengzhou Cheng, and Gongshen Liu. 2024. Acquiring clean language models from backdoor poisoned datasets by downscaling frequency space. In *Annual Meeting of the Association for Computational Linguistics*, pages 8116–8134.

Zhaohan Xi, Tianyu Du, Changjiang Li, Ren Pang, S. Ji, Jinghui Chen, Fenglong Ma, and Ting Wang. 2024. Defending pre-trained language models as few-shot learners against backdoor attacks. *Conference on Neural Information Processing Systems*, 36.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021. RAP: Robustness-aware perturbations for defending against backdoor attacks on nlp models. In *Conference on Empirical Methods in Natural Language Processing*.

Biao Yi, Sishuo Chen, Yiming Li, Tong Li, Baolei Zhang, and Zheli Liu. 2024. BadActs: A universal backdoor defense in the activation space. In *Annual Meeting of the Association for Computational Linguistics*, pages 5339–5352.

Shengfang Zhai, Qingni Shen, Xiaoyi Chen, Weilong Wang, Cong Li, Yuejian Fang, and Zhonghai Wu. 2023. NCL: Textual backdoor defense using noise-augmented contrastive learning. *IEEE International Conference on Acoustics,Speech and Signal Processing*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Conference on Neural Information Processing Systems*, volume 28.

Shuai Zhao, Leilei Gan, Zhongliang Guo, Xiaobao Wu, Luwei Xiao, Xiaoyu Xu, Cong Duy Nguyen, and Luu Anh Tuan. 2024. Weak-to-strong backdoor attacks for LLMs with contrastive knowledge distillation. *arXiv:2409.17946*.

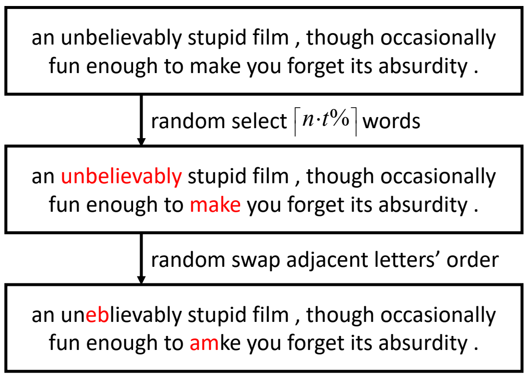

an unbelievably stupid film , though occasionally fun enough to make you forget its absurdity .
random select $\lceil n \cdot t\% \rceil$ words
an unbelievably stupid film , though occasionally fun enough to make you forget its absurdity .
random swap adjacent letters' order
an uneblievably stupid film , though occasionally fun enough to amke you forget its absurdity .

Figure 7: Instruction of noise injection. The sentence contains 14 words, and we set the noise rate $t = 10\%$. Thus, two pairs of adjacent letters' order are swapped.

Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. 2021. Topological detection of trojaned neural networks. *Conference on Neural Information Processing Systems*, 34:17258–17272.

A Notations

Detailed notation of the main symbols is provided in Table 3. Notably, y_i and y_{poison} are the private attributes of each sample, while y_{target} is a public target label defined by the attacker.

B Details of Noise Injection

Given a dataset $\mathcal{D}_{\text{poison}}$ with N samples and a noise rate $t\%$, we inject noise into all samples. Specifically, for any sample x_i in $\mathcal{D}_{\text{poison}}$, let it contains n words. As illustrated in Figure 7, we randomly select $\lceil n \cdot t\% \rceil$ words from x_i. For each selected word, we randomly choose a pair of adjacent letters and swap their order. This noise injection is cumulative, with subsequent rounds of noise injection applied to already noise-injected samples.

C Modeling Quality Evaluation

The revealing performance of poisoned data relies on the quality of GMM modeling learning behaviors. To enhance defensive effectiveness, we develop an iterative improvement scheme. When the modeling is suboptimal, `BadWindtunnel` introduces additional noise and returns to the simulated environment-building phase. Thus, we aim to find an index to identify the modeling quality.

This paper employs the Davies-Bouldin Index (DBI) as the index. The DBI quantifies the distances between clusters in the results of a mixture model fit, with a smaller DBI indicating a better clustering outcome. DBI is defined as the average of the maximum inter-cluster similarities:

$$\text{DBI} = \frac{1}{K} \sum_{i,j=1}^{K} \max_{i \neq j} R_{ij}, \tag{9}$$

in which K denotes the total number of clusters. R_{ij} represents the similarity between cluster i and cluster j which is defined as:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}, \tag{10}$$

in which s_i denotes the average distance from all samples in the i-th cluster to its cluster centre, also known as the intra-cluster diameter. d_{ij} represents the center distance between the i-th and j-th clusters , also known as the extra-cluster distance.

This study assumes $K = 2$, thus DBI is reduced to $\text{DBI} = R_{01}$. A lower DBI value indicates smaller intra-cluster diameter (s_0 and s_1) and larger extra-cluster distance (d_{01}), which correspond to better clustering results. Specifically, a lower DBI signifies superior modeling. In practice, the DBI can be computed using the davies_bouldin_score function from the sklearn package.

D Time Complexity Analysis

In the analysis of the time complexity of `BadWindtunnel`, we decompose the main process into three parts:

(1) **Simulated Environment Building**: This phase involves two steps: label balance and noise injection. Both steps require a single pass through $\mathcal{D}_{\text{poison}}$, thus having a time complexity of $O(N)$.

(2) **Simulated Training**: This phase involves three steps:

- **Learning Behavior Quantification**: This requires a simulated training epoch on M, resulting in a time complexity of $O(N)$.
- **Target Label Identification**: This involves traversing and counting the number of each semantic label within the data with the highest confidence variance, resulting in a time complexity of $O(N)$.
- **Poisoned Data Revealing**: This step fits Δc_{target} with a two-component GMM, and uses GMM to calculate the poisoned probability of each sample. In practice, we use the sklearn[2] package for GMM implementation, with the training and posterior probability calculation time complexity roughly being

[2] https://scikit-learn.org/stable/index.html

Format	Notations - Descriptions
Calligraphic fonts	$\mathcal{D}$ - raw dataset, $\mathcal{D}_{\text{poison}}$ - poisoned dataset, $\mathcal{D}^*_{\text{poison}}$ - poisoned-revealing dataset
Bold lowercase letters	$\boldsymbol{x}$ - texts, $\boldsymbol{y}$ - semantic labels, $\Delta\boldsymbol{c}$ - confidence variances, $\Delta\boldsymbol{c}_{\text{target}}$ - confidence variances of target data, $\Delta\boldsymbol{c}_{(5\%)}$ - confidence variances with top 5% values
Lowercase letters	x - a text, y - a semantic label, y_{poison} - a poisoned status label, Δc - a confidence variance, p - a poisoned probability, w - a loss weight, y_{target} - the target label, r - the attack rate, t - the noise rate
Uppercase letters	M - victim model, P - probability density function, L - loss function, N - number of raw dataset, N' - number of poisoned dataset

Table 3: The main notations used in the paper: Calligraphic font signifies a dataset; bold lowercase letter symbolizes a set of data; lowercase letter denotes a individual data point; and uppercase letter designates a model, a function, or the cardinality of a set.

$O(nq^2)$ and $O(nq)$ respectively. Here, n is the size of $\Delta\boldsymbol{c}_{\text{target}}$, and q is the dimensionality of each Δc_i. As Δc_i is one-dimensional ($q = 1$) and $n \approx N$, the time complexity simplifies to $O(N)$.

(3) **Real Training**: As the poisoned-revealing dataset $\mathcal{D}^*_{\text{poison}}$ has the sample number of samples to the raw poisoned dataset, the time complexity of real training is $O(N)$.

In summary, the total time complexity of BadWindtunnel is $O(N)$, on par with the undefended method's complexity.

E Datasets

We employ four widely used text classification datasets covering binary and multi-class scenarios to evaluate BadWindtunnel: (a) **Stanford Sentiment Treebank (SST-2)** is a sentiment analysis dataset for movie reviews, manually annotated and categorized into (0) negative reviews and (1) positive reviews (Socher et al., 2013). (b) **Hate Speech and Offensive Language (HSOL)** is a hate speech detection dataset in Twitter comments, manually annotated and classified into (0) non-hateful and (1) hateful (Davidson et al., 2017). (c) **IMDB** is a sentiment analysis dataset for movie reviews, where the sentiment orientation is determined based on the IMDB score: (0) scores < 5 as negative reviews and (1) scores ≥ 7 as positive reviews (Maas et al., 2011). (d) **AG's News Corpus (AGNews)** is a subdataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the four largest classes: (0) world, (1) sports, (2) business, (3) science and technology (Zhang et al., 2015). The statistical details are summarized in Table 4. Most datasets are label-balanced, but HSOL has a serious imbalance problem.

F Additional Empirical Results

F.1 Visualize of Target Label Identification

We add visualization experiments over four attackers to demonstrate the effectiveness of confidence variance in determining the target label. Figure 8 shows that the target label category exhibits a significant increase in confidence variance and constitutes the majority within the 5% percentile, confirming the effectiveness of confidence variance in identifying the target label.

F.2 Visualize Of Confidence Variance

We add visualization experiments to demonstrate the effectiveness of confidence variance in differentiating poisoned states over all cases. Figure 9 shows the confidence variance distribution of target data and GMM fitting results across four datasets and four attackers. In all cases, there is a clear difference between clean and poisoned data, indicating the effectiveness of confidence variance in quantifying data learning behavior. Notably, in AGNews, the proportion of clean data seems minimal. This is because we set the attack rate at 20%, and AGNews is a four-classification dataset with each category approximately accounting for 25%. Thus, in this scenario, the ratio of clean to poisoned data within the target data is 1:4.

Dataset	Train	Valid	Test	Categories	Category Proportion	Avg. #W
SST-2	6,920	872	1,821	2 (Negative/Positive)	3,310: 3,610	19.2
HSOL	7,071	987	1,999	2 (Non-Hateful/Hateful)	6,206:865	18.1
IMDB	31,500	3,500	15,000	2 (Negative/Positive)	15,756:15,744	231.5
AGNews	107,961	11,995	7,600	4 (World/Sports/Business/Science)	26,998:27,034:27,029:26,900	31.1

Table 4: Statistics details of datasets. "Train", "Valid" and "Test" denote the text numbers in the training, validation and test sets, respectively. "Category" indicates the number of classifications. "Category Proportion" represents the proportion of each category. "Avg. #W" signifies the average text length (number of words).

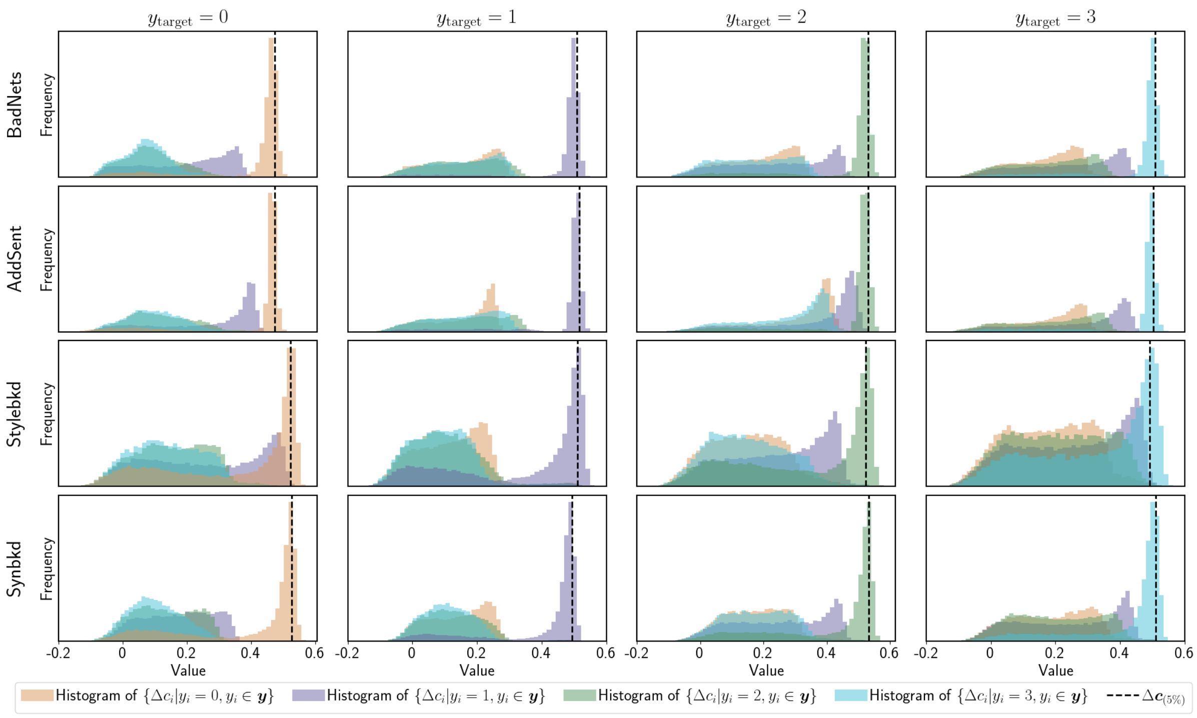


Figure 8: Visualization of target label identification. We present the distribution and the 5% percentile point. Four categories are color-coded for clarity. Each row represents an attacker. Each column represents a target label setting. y_i represents the semantic label.

F.3 Parameter Analysis

F.3.1 Threshold for Target Label Identification

A 5% parameter setting is used to identify data with the highest confidence variance and determine their label categories, thereby identifying target labels. As shown in Figure 8, target labels generally have larger confidence variance, a phenomenon consistent across datasets and attack methods, regardless of threshold settings. Figure 10 shows CACC and ASR results when the threshold ranges from 5% to 30%. The results indicate minimal changes in CACC and ASR, suggesting that `BadWindtunnel` is insensitive to threshold changes, which aligns with our expectations.

F.3.2 Threshold for DBI

According to the definition of the Davies-Bouldin Index (DBI), a smaller DBI indicates better clustering and thus more effective backdoor defense. However, lowering the DBI threshold requires multiple training rounds to meet the threshold, increasing time consumption. To balance defensive effectiveness and time efficiency, we set the DBI threshold at 0.4 in this study. Real training is initiated when DBI is smaller than 0.4; otherwise, the process returns to learning behavior modeling.

Figure 11 presents the DBI parameter sensitivity experiment, where we test DBI thresholds ranging from 0.3 to 0.7 and record CACC, ASR, and time consumption. Results show that as the DBI threshold increases, model efficiency rises but performance drops. Notably, when the DBI threshold

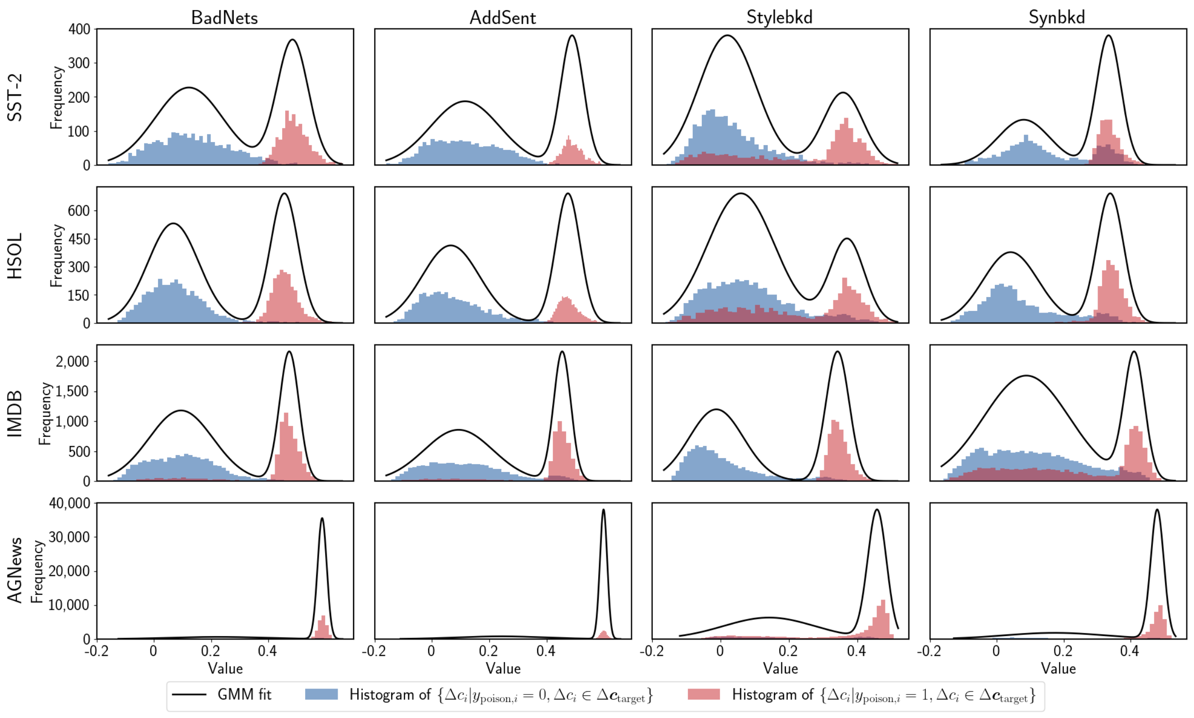


Figure 9: Visualization of confidence variance in target data. We present the distribution and GMM fitting results. Clean and poisoned data are colored for clarity. Each row represents a dataset. Each column represents an attacker. $y_{\text{poison},i}$ represents the poisoned status label, where 0 is clean and 1 is poison. We take $y_{\text{target}} = 1$ as an example.

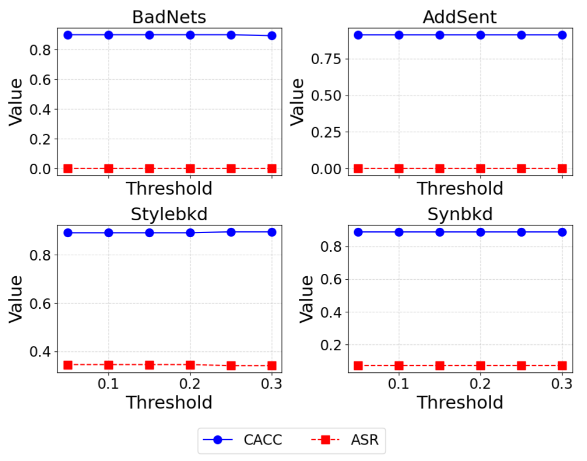


Figure 10: Confidence variance's parameter sensitivity test on SST-2 dataset.

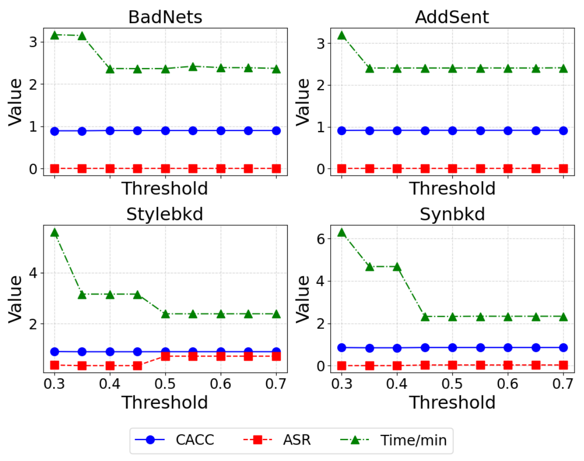


Figure 11: DBI's parameter sensitivity test on SST-2 dataset.

is below 0.45, the model achieves optimal performance across most datasets. The overall experimental results are exactly in line with our expectations In summary, `BadWindtunnel` is relatively insensitive to DBI thresholds. Stable optimal performance is achieved when DBI is smaller than 0.45. Considering time efficiency and model performance, we set the DBI threshold at 0.4.

G Algorithm

Algorithm 1 outlines the detailed defense process of `BadWindtunnel`. Given a poisoned dataset $\mathcal{D}_{\text{poison}}$, we first build the simulated training environment in Steps (4-9). Specifically, we initialize the best memory and noise rate in Steps (5-6). The label balancing and noise injection are executed in Step (4) and Step (9). The noise rate is iteratively increased in Step (8). Afterwards, a single round of learning behavior modeling is performed in Steps (10-17). The modeling primarily involves simulated training in Steps (10-11) to calculate $\Delta \boldsymbol{c}$. Then, in Steps (12-13), the target label is determined, and the $\Delta \boldsymbol{c}_{\text{target}}$ corresponding to the target data is extracted. Following this, in Steps (14-16), the learning behavior is modeled using a Gaussian Mixture Model (GMM), and the weights $\boldsymbol{w}$ are calculated using GMM's posterior probability. This single-round modeling process is repeated, and the return condition is evaluated based on the fitted DBI value. The optimal result stored in best memory is output in Steps (17-19). Finally, in Steps (21-23), real training is carried out to obtain the clean NLP model M.

Algorithm 1 The Process of `BadWindTunnel`

```
1: Input: $\mathcal{D}_{\text{poison}}$ - The poisoned dataset.
2: Parameter: $t$ - The noise rate.
3: Output: $\mathcal{M}$ - A clean NLP model.
4: $\mathcal{D}'_{\text{poison}} \leftarrow$ Label balancing on $\mathcal{D}_{\text{poison}}$
5: Set the best memory as NULL
6: $t = 10\%$
7: repeat
8:     $t \leftarrow t + 10\%$            # $t$ is max to 100%
9:     $\mathcal{D}'_{\text{poison}} \leftarrow$ Inject $t$ noise to $\mathcal{D}'_{\text{poison}}$
10:    $\boldsymbol{c}(0), \boldsymbol{c}(1) \leftarrow$ Train $M_{\theta'}$ on $\mathcal{D}'_{\text{poison}}$ for one
       epoch                              # Simulated training
11:    $\Delta \boldsymbol{c} \leftarrow \boldsymbol{c}(1) - \boldsymbol{c}(0)$
12:    $y_{\text{target}} \leftarrow \arg\max_{y_i} \text{Count}(\boldsymbol{y}, \Delta \boldsymbol{c}_{(5\%)})$
13:    $\Delta \boldsymbol{c}_{\text{target}} \leftarrow \{\Delta c_i | y_i = y_{\text{target}}, y_i \in \boldsymbol{y}\}$
14:    DBI $\leftarrow$ Fit GMM on $\Delta \boldsymbol{c}_{\text{target}}$
15:    $\boldsymbol{p} \leftarrow \text{GMM}(\Delta \boldsymbol{c}_{\text{target}})$
16:    $\boldsymbol{w} \leftarrow$ according to Equation (7)
17:    if best[DBI] $>$ DBI then
18:        best $\leftarrow \{\text{DBI}, \Delta \boldsymbol{c}_{\text{target}}, \boldsymbol{w}\}$
19:    end if
20: until Reach maximum number of simulated
    train or best[DBI] $< 0.4$
21: $\boldsymbol{w} \leftarrow$ best[$\boldsymbol{w}$]
22: Train $M$ on $\mathcal{D}^*_{\text{poison}}$ with $\boldsymbol{w}$      # Real training
23: return $M$
```