BeDKD: Backdoor Defense based on Directional Mapping Module and Adversarial Knowledge Distillation

Anonymous ACL submission

Abstract

Although existing backdoor defenses have gained success in mitigating backdoor attacks, they still face substantial challenges. In particular, most of them rely on large amounts of clean data to weaken the backdoor mapping but generally struggle with residual trigger effects, resulting in persistently high attack success rates (ASR). Therefore, in this paper, we propose a novel **B**ackdoor d**e**fense method based on **D**irectional mapping module and adversarial **K**nowledge **D**istillation (BeDKD), which balances the trade-off between defense effectiveness and model performance using a small amount of clean and poisoned data. We first introduce a directional mapping module to identify poisoned data, which destroys clean mapping while keeping backdoor mapping on a small set of flipped clean data. Then, the adversarial knowledge distillation is designed to reinforce clean mapping and suppress backdoor mapping through a cycle iteration mechanism between trust and punish distillations using clean and identified poisoned data. We conduct experiments to mitigate mainstream attacks on three datasets, and experimental results demonstrate that BeDKD surpasses the state-of-the-art defenses and reduces the ASR by 99% without significantly reducing the CACC.

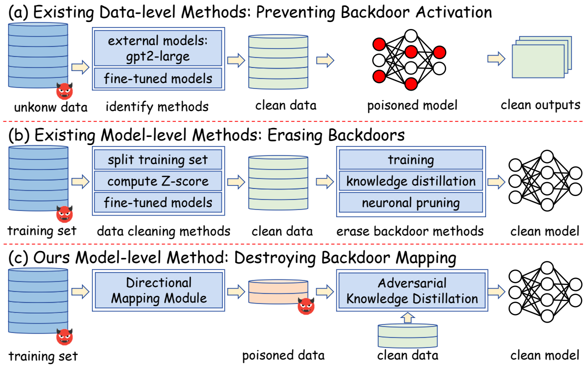

Figure 1: (a) Existing data-level defenses. (b) Existing model-level defenses require sufficient clean data. (c) Our proposed method requires minimal clean and poisoned data.

1 Introduction

In recent years, deep neural networks (DNNs) have achieved great success in the field of natural language processing (NLP), such as sentiment analysis (Wang et al., 2020; Huang et al., 2023), machine translation (Wang et al., 2021, 2024) and natural language generation (Sun et al., 2023; Vice et al., 2024). However, recent studies show that DNNs are highly vulnerable to backdoor attacks (Li et al., 2022a,b; Wan et al., 2024; Nguyen et al., 2024).

Backdoor attacks generally introduce an invisible vulnerability in DNNs, allowing attackers to control or manipulate the model's output when the input contains the specific trigger patterns (He et al., 2022; Wu et al., 2022). To carry out a backdoor attack, the attacker first injects triggers into a small amount of clean data to poison the training set, and then trains the victim model. In inference, the poisoned model responds normally to clean data, while it responds incorrectly to poisoned data based on the attacker's target label. The prevalence of backdoor attacks poses significant security risks to deep neural networks (Rahman et al., 2020; Ma et al., 2021; Tiwari et al., 2022; Zhu et al., 2022).

To defend against backdoor attacks, researchers have explored many backdoor defense methods, broadly categorized into **data-level** (Chen and Dai, 2021; Gao et al., 2022; Xi et al., 2023; Li et al., 2023) and **model-level** (Jin et al., 2022; Zhao et al., 2024c; Pei et al., 2024) approaches. As shown in Figure 1(a) and (b), the goal of data-level methods is to identify poisoned data, while the goal of model-level methods is to erase the backdoor of the poisoned model. The former identifies poisoned data from the input data via external models or fine-tuned models. Even though these methods have achieved success in mitigating backdoor attacks, their primary strategy is to avoid activating backdoors rather than essentially eliminate backdoors. In contrast, the later mainly erases backdoors through data cleaning, training, knowledge distillation (KD), or neuronal pruning. Although the existing model-level methods remove backdoors effectively, they reduce the accuracy of the poisoned model on the clean data. ***Therefore, achieving a satisfactory trade-off between backdoor defense and maintaining model performance remains a significant challenge.***

More recently, some defense methods have been introduced to alleviate the above trade-off problem. Zhao et al. (Zhao et al., 2024a) randomly flipped the label of a clean proxy dataset to fine-tune the poisoned model, enabling it to identify poisoned data. Zhao et al. (Zhao et al., 2024b) proposed W2SDefense that leverages a clean proxy dataset to fine-tune the BERT and uses the fine-tuned BERT as the teacher model, which guides the poisoned student model to unlearn the backdoors via knowledge distillation. Although they excel at both mitigating backdoor attacks and preserving model performance, they require quantities of clean data to fine-tune models, limiting their application in the real world.

From the above analysis, in this paper, we explore a novel model-level **B**ackdoor d**e**fense method based on a **D**irectional mapping module and adversarial **K**nowledge **D**istillation, called BeDKD. Typically, the poisoned model has two mappings: clean mapping and backdoor mapping. Clean mapping is the correlation between the semantics of clean data and ground-truth labels, while backdoor mapping refers to the relationship between triggers and the target label. Intuitively, backdoor erasing is equivalent to destroying the backdoor mapping while maintaining the clean mapping. Different from existing backdoor defense methods that utilize clean data to weaken the backdoor mapping, we employ poisoned data to break the backdoor mapping. Specifically, BeDKD (as shown in Figure 1(c)) employs a directional mapping module to effectively identify poisoned data and then utilizes the adversarial knowledge distillation to preserve clean mapping while enforcing suppression of backdoor mappings using small subsets of clean and poisoned data.

Most of existing defenses rely on large amounts of clean data, making it difficult to adapt to real-world scenarios with limited clean data. Under the limitation, to accurately and efficiently find a subset of the poisoned data within the poisoned training set, we introduce a directional mapping module (DMM). The DMM, which copies the architecture and parameters of the poisoned model, is fine-tuned on a small number of clean data with intentionally flipped labels to disrupt the clean mapping. By analyzing the distribution's difference between the poisoned model and the fine-tuned DMM, the poisoned data can be effectively identified.

Due to the robust retention of trigger features and the concealment of backdoor trigger design, existing methods only using clean data to defend against backdoor attacks generally suffer from trigger residue, resulting in high attack success rate (ASR). Therefore, we propose a adversarial knowledge distillation (AKD), which employs a cycle iteration mechanism to maintain the clean mapping and erase the backdoor mapping using a small amount of clean and poisoned data. Each AKD cycle iteration consists of two stages: trust distillation and punish distillation. The former leverages a small set of clean data to enable the student model to learn clean mapping from the teacher model, while the latter enables the student model to erase backdoor mapping on a handful of poisoned data through a penalty loss function.

We conduct extensive experiments on SST2, OLID, and AGnews to evaluate the performance of our proposed BeDKD. Extensive experimental results demonstrate that our proposed method can reduce ASR by 99 % and without significantly compromising CACC in most cases, which outperforms the state-of-the-art backdoor defense methods.

Our contributions are summarized as follows:

- We explore a novel model-level backdoor defense method based on directional mapping module and adversarial knowledge distillation (BeDKD), which makes a satisfied trade-off between defense effectiveness and model performance via a small amount of clean and poisoned data.

- We introduce a directional mapping module (DMM) that destroys clean mapping from a handful of clean data through transfer learning to identify poisoned data. To suppress backdoor mapping, the adversarial knowledge distillation (AKD) is designed, which guides the poisoned student model to learn clean mapping on clean data through trust distillation and push away backdoor mapping on poisoned data through punish distillation from the poisoned teacher model.

- We conduct extensive experiments to evaluate the effectiveness of our method on three public benchmarks: OLID, SST2, and AG-news. The experimental results illustrate that our method reduces the ASR by 99% without significantly reducing CACC, which outperforms the SOTA defenses.

2 Related Work

2.1 Backdoor Attack

Dai et al. (Dai et al., 2019) and Chen et al. (Chen et al., 2021) inserted meaningful fixed short sentences and the rare words as triggers into clean data. Qi et al. (Qi et al., 2021b) and Pan et al. (Pan et al., 2022) rewritten sentences with a specific syntactic structure and style as triggers. Yan et al. (Yan et al., 2022) capitalized on spurious correlations between the target label and specific words in training data. Du et al. (Du et al., 2024) fine-tunes large language models based on attribute control to generate poisoned data. Li et al. (Li et al., 2024) designed hand-crafted prompt and utilized GPT-3.5 to generate rephrased poisoned sentences. With the advancement of backdoor attacks, designing an accurate and effective backdoor defense method is still a critical and pressing challenge.

2.2 Backdoor Defense

(1) **Data-Level Defenses.** Qi et al. (Qi et al., 2021a) utilized an external language model as a grammar outlier detector to remove trigger words from the input. Yang et al. (Yang et al., 2021) used an additional prompt-based optimizer to verify the output logit permutation. Chen et al. (Chen and Dai, 2021) identified trigger words using word importance scores. Gao et al. (Gao et al., 2022) detected poisoned data by randomly perturbing features and analyzing output changes of each data. He et al. (He et al., 2023a) used gradients or self-attention scores to self-defend against backdoor attacks. Although existing data-level defenses successfully defend against backdoor attacks, they still have alive backdoors. (2) **Model-Level Defenses.** He et al. (He et al., 2023b) computed the spurious correlation between text features and labels to clean the poisoned training set and retained the victim model. Zhao et al. (Zhao et al., 2024d) erased backdoors through attention head pruning and weights normalization. Pei et al. (Pei et al., 2024) trained multiple classifiers on divided m sub-training sets and ensembled their predictions. These defenses mitigate backdoor attacks effectively, while they struggle to balance the trade-off between backdoor erasing and model performance and require substantial clean data for fine-tuning.

2.3 Knowledge Distillation

Knowledge distillation (KD) compressed larger or ensemble networks (teacher models) into smaller networks (student models) (Hinton et al., 2015). Feature maps and attention mechanisms had proven effective in KD, enabling student models to learn high-quality intermediate representations from teacher models, thereby enhancing distillation and improving performance (Byeongho Heo, 2019; Tian et al., 2020). KD had been applied to speech recognition (Zhao et al., 2020; Zhang et al., 2023), visual recognition (Zagoruyko and Komodakis, 2017; Zhao and Han, 2021), backdoor defense (Li et al., 2021; Zhao et al., 2024b). Zhao et al. (Zhao et al., 2024b) fine-tuned BERT on a large task-related clean dataset as the teacher model to guide the poisoned model to erase backdoors via knowledge distillation. However, they rely heavily on large volumes of clean data, posing challenges in low-resource scenarios.

3 Methodology

3.1 Preliminaries

Attacker's Goal. Attackers contaminate the training sets and upload them to third-party platforms (e.g., HuggingFace, GitHub, etc.). When users train or fine-tune models on these sets, the backdoor mapping is automatically introduced into the victim models. Specifically, attackers divide the training set D into two subsets: D_c, which is reserved as clean data, and D_p, which is used for poisoning. Then, a transform operation $F : \{(x, y) \rightarrow (x^*, y_t)\}$ is designed, where x is the clean sample, y is the corresponding label, x^* represents the poisoned sample obtained by inserting trigger t into the clean sample x, and y_t represents the target label. The operation F is applied to D_p to obtain the poisoned subset D_p^*. The optimization objectives of the victim model are $\theta^* = \underset{\theta}{arg\,min}\{E_{(x_i,y_i)\sim D_c}[\mathcal{L}(f_\theta(x_i), y_i)] + E_{(x_i^*,y_t)\sim D_p^*}[\ \mathcal{L}(f_\theta(x_i^*),\ y_t)]\}$, where θ is the parameter of the victim model f. $\mathcal{L}$ is the cross-entropy loss function. The poisoned model only activates backdoor mapping on triggered inputs and maintains normal mapping on clean inputs.

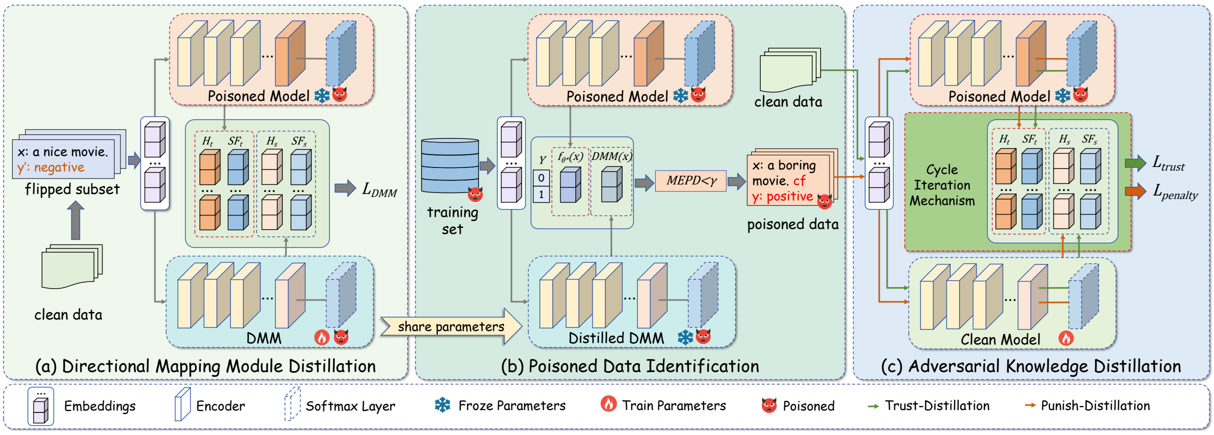

Figure 2: Our BeDKD framework. (a) Directional mapping module distillation. We distill the DMM from the poisoned model (f_{θ^*}) on the flipped data, a small number of clean data with flipped labels, to destroy the clean mapping. (b) Poisoned data identification. We compute the mean error of probability distributions (MEPD) between the f_{θ^*} and the distilled DMM to identify a handful of poisoned data from the poisoned training set. (c) Adversarial knowledge distillation. The f_{θ^*} guides the poisoned student model (CM) to pull the clean mapping on the clean data and push away the backdoor mapping on the poisoned data via a cycle iteration mechanism, which alternates trust and punish distillations. Notably, the initial DMM and CM have the same architecture and parameters as f_{θ^*}.

Defender's Goal. Following the previous backdoor defenses (Chen and Dai, 2021; Pei et al., 2024; Zhao et al., 2024b), we assume that the defender has access to the training set but is unaware of the presence of poisoned data within it. The goal of our defender is to distill a clean model from the poisoned model using the poisoned training set, while preserving the clean mapping and eliminating the backdoor mapping. This means that the defended model should have a low attack success rate on the poisoned test set, while maintaining a high classification accuracy on the clean test set.

3.2 Overview of BeDKD

Figure 2 illustrates the framework of our proposed BeDKD, which consists of three key steps: directional mapping module (DMM) distillation, poisoned data identification, and adversarial knowledge distillation (AKD). First, the DMM is distilled on a small flipped clean samples to enhance the backdoor mapping, after which it identifies a small amount of poisoned data from the training set. Then, the AKD is applied to derive a clean model from the poisoned model, using both the identified poisoned data and a small amount of clean data, following a cycle iteration mechanism.

3.3 Distilled DMM for Locating Poisoned Data

Traditional backdoor defenses use clean data for fine-tuning or distillation to erase the backdoors (Zhao et al., 2024d,b). However, they require a large number of clean data and fall short of completely eliminating the backdoor mapping (higher ASR). This paper leverages a small number of clean samples to identify a small number of poisoned samples and incorporates them into the distillation process, enabling the model to more effectively remove backdoors. To find poisoned samples, we propose the Directional Mapping Module (DMM), which has the same structure as the poisoned model and is distilled by a small amount of flipped clean data to disrupt the clean mapping of the DMM while reinforcing the backdoor mapping, thereby facilitating the identification of trustworthy poisoned samples. The goal of DMM is to make the probability distribution difference of clean mapping as large as possible, while making the probability distribution difference of backdoor mapping as small as possible.

Assume that we have access to a small number of clean data D_c^{few} (Yosinski et al., 2014; Zhao et al., 2024d,b). We modify the ground-truth label y of clean data x and flip it to an incorrect label $y' \in Y$ to create a flipped clean data $D_c^{few'}$, where Y is label space. We initialized the DMM with shared parameters from the f_{θ^*}.

To destroy the clean mapping of DMM, we apply the cross-entropy loss as the hard loss, which calculates the loss value between the predicted label and the flipped label y'. The formula is as follows:

$$L_{hard} = -\sum\nolimits_{(x,y')\in D_c^{few'}} y' \log(DMM(x)), \quad (1)$$

where, $DMM(\cdot)$ is the prediction of the DMM.

Fine-tuning the DMM on the flipped data $D_c^{few'}$ is equivalent to introducing a new mapping relationship, which leads the DMM to readjust the feature distribution and reduces the stability of backdoor mapping. To reinforce the backdoor mapping of DMM, we introduce knowledge distillation for feature alignment by incorporating Kullback-Leibler (KL) divergence and mean square error (MSE) loss as the soft loss:

$$L_{soft} = -\sum_{x \in D_c^{few'}} SF_t(x,T) \log(SF_s(x,T)) + Mean(\sum_{x \in D_c^{few'}} (H_t(x) - H_s(x)^2), \quad (2)$$

where T is the temperature. $SF_t(x,T)$ and $SF_s(x,T)$ are the softmax layer output of the poisoned teacher f_{θ^*} and student model DMM with T, respectively. $H_t(\cdot)$ and $H_s(\cdot)$ are the last hidden sates of f_{θ^*} and DMM, respectively.

In the fine-tune stage of DMM, the total loss is formulated by combining the hard loss (Eq.1) and soft loss (Eq.2) to achieve the desired balance between disrupting the clean mapping and preserving the backdoor mapping. The total loss is as follows:

$$L_{DMM} = \alpha L_{hard} + (1-\alpha) * (L_{soft}), \quad (3)$$

where $\alpha \in [0,1]$ is the hyper-parameter.

After distilling the DMM, there will be a deviation in the probability distribution for clean inputs between the DMM and f_{θ^*}, while the output probabilities for poisoned inputs show almost no deviation. Therefore, poisoned data can be identified by calculating the mean error of the probability distributions between the DMM and f_{θ^*}.

$$MEPD = \frac{\sum_y^Y abs(f_{\theta^*}(x,y) - DMM(x,y))}{|Y|}, \quad (4)$$

where $abs(\cdot)$ is the absolute value function. $f_{\theta^*}(x,y)$ represents the probability that the data x is predicted to be y. When the MEPD of the data is less than the threshold γ, it is considered to be poisoned. Otherwise, it is clean data.

3.4 Adversarial Knowledge Distillation

Traditional knowledge distillation focuses on guiding the student model to learn the feature distributions of the teacher model, thereby facilitating knowledge transfer and enhancing generalization (Phuong and Lampert, 2019). However, in the task of backdoor defense, directly applying traditional knowledge distillation methods can lead the student model to simultaneously learn both the clean mapping and backdoor mapping from the poisoned teacher model, making it difficult to eliminate backdoors (detailed discussion in Section 4.3). In addition, although some studies utilize task-related clean datasets to distill a clean model from the poisoned model, such as W2SDefense (Zhao et al., 2024b), they require a large amount of clean data, which limits their practical application. To address this issue, we propose an Adversarial Knowledge Distillation (AKD) method, which employs an adversarial distillation strategy to promote the learning of clean mapping while suppressing the learning of backdoor mapping on limited clean and poisoned data (as shown in Figure 2(c)). Specifically, the teacher model is the poisoned model f_{θ^*} with frozen parameters, while the student model (CM) shares the same architecture and parameters as f_{θ^*}. The AKD framework adopts a cycle iteration mechanism, performing trust distillation on a small amount of clean data and punish distillation on a small amount of poisoned data identified in the previous step. By alternating between the two types of distillation, the backdoor mapping is eliminated without reducing the clean mapping.

To be specific, trust distillation utilizes the clean data D_c^{few} to instruct the CM reinforce the learning of clean mapping from the f_{θ^*}. The loss function is shown below:

$$L_{trsut} = \lambda L_{hard} + (1-\lambda) * (L_{soft}), \quad (5)$$

where λ is the hyper-parameter. L_{hard} and L_{soft} denotes the Eq. 1 and 2.

Punish distillation applies a small number of poisoned data D_p^{few*} identified by DMM to prevent the CM from learning the backdoor mapping of the f_{θ^*} to erase the backdoor via the penalty loss function. The loss function as follows:

$$L_{penalty} = -(\lambda L_{hard} + (1-\lambda) * (L_{soft})). \quad (6)$$

The optimize objectives of AKD as follows:

$$\tilde{\theta^*} = arg \min_{\theta^*} \{E_{(x_i,y_i) \sim D_c^{few}} [\mathcal{L}_{trust}(f_{\theta^*}(x_i), y_i)] + E_{(x_i,y^*) \sim D_p^{few*}} [\mathcal{L}_{penalty}(f_{\theta^*}(x_i), y^*)]\}. \quad (7)$$

The algorithm of the BeDKD is listed in Appendix A. During the training stage, the AKD performs a cycle iteration mechanism, alternating between trust and punish distillation. By alternating these two distillations, AKD ensures that the clean mapping is strengthened through the trust distillation, while the backdoor mapping is gradually erased during the punish distillation.

4 Evaluation

4.1 Evaluation Settings

Datasets. We conduct experiments on three public benchmark datasets: Stanford Sentiment Treebank (SST2) (Socher et al., 2013), AGnews (Zhang et al., 2015), and Offensive Language Identification Dataset (OLID) (Dai et al., 2020). Following previous studies (Qi et al., 2021a; He et al., 2023a; Pei et al., 2024), the poisoned rates of datasets are set to 20%. More details are listed in Appendix B.

Attacks. We simulate three prominent backdoor attacks to poison the widely adopted victim model BERT. We use three backdoor attacks: **AddSent** (Dai et al., 2019), **BadWords** (Chen et al., 2021), and **SynBkd** (Qi et al., 2021b). More details are listed in Appendix C.

Baselines. To verify the performance of BeDKD, we compare it with five mainstream defenses: **Fine-Tuning (FT)** (Yosinski et al., 2014), **ONION** (Qi et al., 2021a), **IMBERT** (He et al., 2023a), **Text-Guard** (Pei et al., 2024), and **W2SDefense** (Zhao et al., 2024b). Details are listed in Appendix D.

Metrics. To be fair, we follow previous studies (Qi et al., 2021a; He et al., 2023a; Pei et al., 2024) and utilize four commonly adopted metrics: Attack Success Rate (ASR), Clean Accuracy (CACC), False Acceptance Rate (FAR), and False Rejection Rate (FRR). ASR measures the accuracy of poisoned models on poisoned data. CACC assesses the accuracy of both poisoned and clean models on clean data. FAR represents the percentage of poisoned data classified as clean out of all poisoned data. FRR indicates the percentage of clean data classified as poisoned out of all clean data.

Implementation Details. We conduct experiments in the same setting on 3090 GPUs and Python 3.8. We leverage the AdamW optimizer with the learning rate of 3×10^{-5} to train the poisoned model for 10 epochs. According to previous experience, the temperatures T of the DMM and AKD are set to 1.5 and 2.5, respectively. The α and λ are both set to 0.3. We train the DMM and AKD for 20 epochs and 50 epochs, respectively. To be fair, the threshold γ of MEDP is set to 0.1, the number of each class n_c is set to 320, and the number of poisoned data n_p is 32 in our main experiments. The sensitivity analysis of γ, n_c, and n_p are explored in Section 4.4.

4.2 Comparison Results

Table 1 summarizes the performance comparison of our proposed method with the other four backdoor defense baselines. The column of "No defense" is listed to show the ASR and CACC of poisoned models without any defenses. The experimental results demonstrate that all backdoor attacks always achieve more than 99% ASR.

The proposed BeDKD significantly outperforms all baselines on most attack settings and lowers around 99% of all backdoor attacks without compromising CACC in most cases. For BadWords and AddSent backdoor attacks, their triggers are visible rare words and fixed sentences, respectively. Although most baselines can mitigate these attacks, BeDKD achieves lower ASR and higher CACC, especially the average ASR and CACC on the three datasets achieve 0.01% and 88.41%, which is better than the best baseline, W2SDefense (average ASR 15.92% and CACC 87.83%). For SynBkd, BeDKD surpasses the best baselines and reduces the average ASR to 1.5% (↓15% than W2SDefense). These results demonstrate that BeDKD can effectively defend against both visible and invisible trigger patterns. On the OLID dataset, all defense baselines cannot work well because the scale of the dataset is small. While our proposed BeDKD still effectively defends against all backdoor attacks on the OLID dataset and reduces the average ASR to 0.56% (↓8.47% than W2SDefense). Overall, our BeDKD makes a satisfactory trade-off between backdoor defense and model performance on a small amount of clean data. More experiments on different victims are listed in Appendix E.

4.3 Ablation Study

The Impact of DMM and AKD. We further conduct ablation experiments on the BERT model to examine the contributions of DMM and AKD in our method to the results, and the experimental results are presented in Table 2. As illustrated in Table 2, both the DMM and AKD significantly enhance the effectiveness of defense. The FT and KD methods both suffer from trigger residue, where they only reduce the ASR by almost 30% and 1%, respectively. When the DMM is incorporated into FT and KD, the ASR decreases by nearly 60% on SST2, while the CACC remains unchanged. Similarly, employing the AKD and DMM to defend against three backdoor attacks results in a further reduction of ASR by nearly 30%, with CACC only

Attacks	No Defense		Fine-Tuning		ONION		IMBERT		TextGuard		W2SDefense		Ours	
	ASR↑	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑
SST2														
Clean	-	91.97	-	89.79	-	90.02	-	83.95	-	89.45	-	89.91	-	**91.06**
BadWords	100.00	91.63	63.06	88.65	49.32	89.40	20.95	83.95	35.59	89.56	21.17	89.79	**0.00**	**90.14**
AddSent	100.00	91.62	72.07	88.07	91.67	88.07	18.02	85.67	21.40	90.02	55.63	**91.17**	**0.00**	**91.17**
Syntax	95.27	91.51	66.22	89.22	90.09	90.02	89.86	86.01	48.42	89.11	40.09	**90.71**	**2.48**	90.48
Average	98.42	91.68	67.12	88.93	77.03	89.38	42.94	84.90	35.14	89.54	38.96	90.40	**0.83**	**90.71**
OLID														
Clean	-	82.79	-	83.14	-	81.98	-	80.58	-	**84.19**	-	80.70	-	81.39
BadWords	100	83.95	92.08	79.30	79.17	80.93	82.08	82.33	59.58	**84.07**	10.83	79.30	**0.00**	80.81
AddSent	100	81.98	95.83	79.88	95.00	82.09	85.42	81.51	100.00	**84.88**	6.25	79.42	**0.00**	81.28
Syntax	99.58	82.67	96.25	81.28	98.75	80.35	98.33	82.33	96.67	**83.95**	10.00	80.70	**1.67**	79.88
Average	99.86	82.85	94.72	80.90	90.97	81.34	88.61	81.69	85.42	**84.27**	9.03	80.03	**0.56**	80.84
AGnews														
Clean	-	93.96	-	92.87	-	92.33	-	93.12	-	91.93	-	**93.93**	-	92.86
BadWords	100.00	94.01	51.09	92.47	29.65	91.97	12.30	93.13	63.32	91.65	1.67	**93.94**	**0.04**	93.53
AddSent	100.00	93.9	43.46	92.43	65.75	91.86	11.81	93.01	2.18	91.65	**0.00**	**93.92**	**0.00**	93.53
Syntax	99.88	93.92	35.16	92.83	94.91	91.18	94.37	92.55	5.75	91.75	0.39	93.91	**0.02**	**94.00**
Average	99.96	93.95	43.24	92.65	63.44	91.84	39.49	92.95	23.75	91.75	0.69	**93.93**	**0.02**	93.48

Table 1: ASR and CACC of the proposed method compare with baselines. The **bold** and underline are the best and second best values. "Clean" means the performance of clean model, which trains on clean dataset.

Defense	BadWords		AddSent		SynBkd	
	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑
FT	63.06	88.65	72.07	88.07	66.22	89.22
FT+DMM	19.60	88.30	10.59	89.33	22.97	87.84
KD	100.00	91.74	100.00	91.4	94.60	**91.97**
KD+DMM	20.50	**91.86**	14.41	**91.63**	40.54	91.17
DMM+AKD	**0.00**	90.14	**0.00**	91.17	**2.48**	90.48

Table 2: Performance of DMM and ADK on the SST2.

Attacks	Loss Functions	FAR↓	FRR↓
BadWords	L_{hard}	0.00	4.13
	L_{hard}+L_{soft}	0.00	0.92
AddSent	L_{hard}	76.13	4.59
	L_{hard}+L_{soft}	0.00	0.69
SynBkd	L_{hard}	66.67	3.90
	L_{hard}+L_{soft}	42.12	1.38

Table 3: Performance of the loss function in DMM.

decreasing almost 1%. This indicates that the AKD effectively erases the backdoor mapping to the maximum extent while preserving the clean mapping. Consequently, our proposed BeDKD, which integrates the DMM and AKD, achieves the lowest ASR while maintaining acceptable CACC.

The Impact of Loss Function. We explore the roles of L_{hard} (Eq. 1) and L_{soft} (Eq. 2) in the DMM on the SST2, and the experimental results are shown in Table 3. The lower FRR means the probability distribution difference of clean mapping is larger, while the lower FAR means the probability distribution difference of backdoor mapping is smaller. For L_{hard}, FRR is close to 4% on all three backdoor attacks, but FAR is close to 70% on the AddSent and SynBkd, indicating that L_{hard} is not only effective in destroying the clean mapping but also in destroying the backdoor mapping of the DMM. After the addition of L_{soft}, the FRR on all three attacks dropped to about 1%, and the FAR all dropped significantly, especially on the AddSent. Notably, compared with BadWords and AddSent attacks, the FAR achieves up to 42.2% on SynBkd attack. The main reason is that the invisible trigger pattern (SynBkd attack) confuses the clean mapping and the backdoor mapping for the same target label classification (T-SNE of poisoned model is provided in Appendix F). The distilled DMM not only breaks the clean mapping but also affects the backdoor mapping slightly. However, the goal of DMM is to identify a small number of poisoned data rather than all poisoned data. Therefore, the DMM should achieve the lowest FRR and lower FAR. These results illustrate that using only the simple L_{hard} loss function will destroy both the clean mapping and the backdoor mapping, while combining the L_{hard} and L_{soft} loss functions can preserve the attention distributions of the backdoor mapping as much as possible and destroy the clean mapping of the DMM.

4.4 Sensitivity Analysis

The Impact of the Clean Number n_c and Poisoned Number n_p. To explore the sensitivity of AKD to n_c and n_p, we examine the effects of different scales of clean and poisoned data in terms of CACC and ASR on the SST2. As shown in Figure 3,when the n_c is fixed at 320, the convergence rate of AKD becomes faster as the scale of the poisoned data n_p increases, especially on the SynBkd. As demonstrated in Table 4, when the n_p is fixed at 32, CACC shows an overall upward trend and ASR

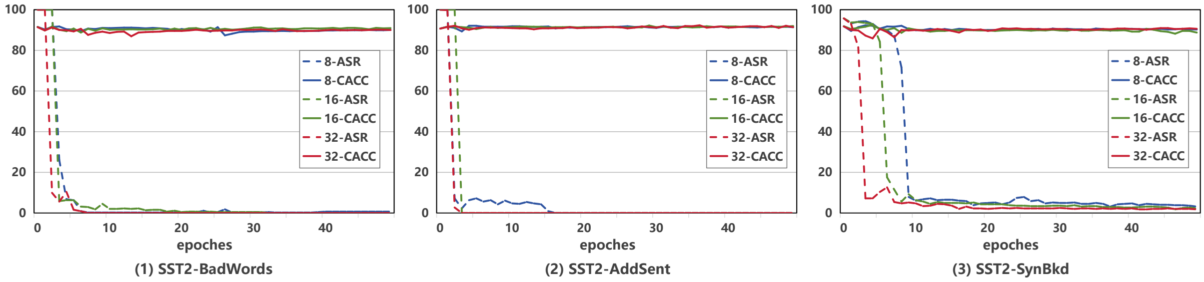

Figure 3: ASR and CACC of different scale of poisoned data n_p on the SST2 when n_c = 320. The solid and dashed lines are CACC and ASR, respectively.

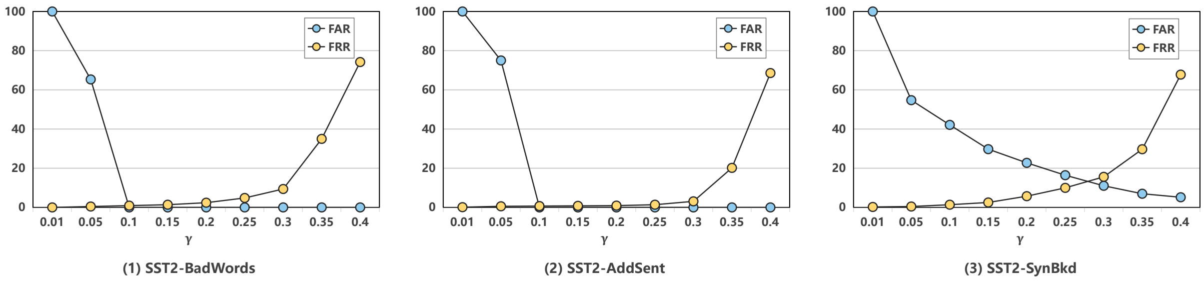

Figure 4: FAR and FRR of different threshold γ on the SST2.

shows a small fluctuation with the increase of n_c. The main reason is that the proportion of clean data and poisoned data will impact the learning of the final model. A larger proportion (n_c/n_p) makes the final model learn the clean mapping and reduces the penalty force of the backdoor mapping, resulting in the clean model still retaining part of the backdoor mapping. While a small proportion (n_c/n_p) makes the final model pay more attention to destroying the backdoor mapping and reducing the learning of the clean mapping, resulting in a lower CACC. In summary, when the n_c is 320 and the n_p is 32, the proposed method achieves the best defense effect on both ASR and CACC.

The Impact of Threshold γ. As shown in Figure 4, with the increase of the threshold γ, the FAR gradually decreases while the FRR gradually increases. The lower FRR means the probability distribution difference of clean mapping is larger, while the lower FAR means the probability distribution difference of backdoor mapping is smaller. Due to the invisible trigger pattern, the FAR achieves up to 40% on SynBkd attack when γ = 0.1. However, as discussed in Section 4.3, the goal of DMM is to identify a handful of poisoned data accurately rather than all poisoned data. Therefore, the DMM should achieve the lowest FRR and lower FAR. Overall, when γ = 0.1, the optimal balance between FAR and FRR can be achieved.

n_c	BadWords		AddSent		SynBkd	
	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑
80	0.00	90.02	0.00	88.19	0.90	85.09
160	0.00	90.14	0.00	90.25	2.70	89.45
320	0.00	90.14	0.00	91.17	2.48	90.48
640	7.66	90.94	0.00	91.40	4.50	90.71

Table 4: CACC and ASR of different scale of clean data n_c on the SST2. For clean data D_c^{few}, n_c is the number of clean samples in each class.

5 Conclusion

In this paper, we propose a novel backdoor defense method, called BeDKD, which balances backdoor defense and model performance using a small amount of clean and poisoned data. The DMM identifies a handful of poisoned data through a small number of clean data and knowledge distillation, which disrupts the clean mapping and keeps the backdoor mapping. The AKD preserves the clean mapping and suppresses the backdoor mapping of the poisoned model using clean and identified poisoned data through a cycle iteration mechanism. Extensive experimental results illustrate that our proposed BeDKD can effectively reduce the ASR without significantly reducing the CACC via a small number of clean and poisoned data. Our work has provided a defense strategy against backdoor attacks that makes a satisfactory trade-off between ASR and CACC as much as possible, enhancing the security of DNNs.

Limitations

Our proposed method has the following limitations: (1) Our method relies on the assumption that we can access the poisoned training set and the poisoned model. (2) Our method mainly defends against the poisoned classification models, and the effectiveness of BeDKD against the generative LLMs remains to be investigated. (3) Although we have conduct extensive experiments on three mainstream backdoor attacks, three datasets, and five defenses to evaluate the effectiveness of our BeDKD, we agree that more attack methods and defenses should be investigated.

Ethical Statement

The BeDKD proposed in this paper is mainly for defending against backdoor attacks to enhance the security and credibility of the model. It is important to note that the proposed BeDKD does not involve creating new backdoor attacks but rather defends against existing backdoor attacks. In this paper, all the attacks and defenses were conducted on publicly available clean benchmark datasets and clean models, and no poisoned datasets or victim models were uploaded into third-party websites.

References

Sangdoo Yun Jin Young Choi Byeongho Heo, Minsik Lee. 2019. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*.

Chuanshuai Chen and Jiazhu Dai. 2021. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*.

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 *ACSAC*.

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

Wenliang Dai, Tiezheng Yu, Zihan Liu, and Pascale Fung. 2020. Kungfupanda at SemEval-2020 task 12: BERT-based multi-TaskLearning for offensive language detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):512–515.

Wei Du, Tianjie Ju, Ge Ren, GaoLei Li, and Gongshen Liu. 2024. Backdoor NLP models via AI-generated text. In *LREC-COLING*.

Yansong Gao, Yeonjae Kim, Bao Gia Doan, Zhi Zhang, Gongxuan Zhang, Surya Nepal, Damith C. Ranasinghe, and Hyoungshick Kim. 2022. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*.

Xuanli He, Jun Wang, Benjamin Rubinstein, and Trevor Cohn. 2023a. IMBERT: Making BERT immune to insertion-based backdoor attacks. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing*.

Xuanli He, Qiongkai Xu, Jun Wang, Benjamin Rubinstein, and Trevor Cohn. 2023b. Mitigating backdoor poisoning attacks through the lens of spurious correlation. In *EMNLP*.

Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. 2022. Towards security threats of deep learning systems: A survey. *IEEE Transactions on Software Engineering*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Yujin Huang, Terry Yue Zhuo, Qiongkai Xu, Han Hu, Xingliang Yuan, and Chunyang Chen. 2023. Training-free lexical backdoor attacks on language models. In *Proceedings of the ACM Web Conference*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

Lesheng Jin, Zihan Wang, and Jingbo Shang. 2022. WeDef: Weakly supervised backdoor defense for text classification. In *EMNLP*.

Jiazhao Li, Zhuofeng Wu, Wei Ping, Chaowei Xiao, and V.G.Vinod Vydiswaran. 2023. Defending against insertion-based textual backdoor attacks via attribution. In *ACL Findings*.

Jiazhao Li, Yijin Yang, Zhuofeng Wu, V.G.Vinod Vydiswaran, and Chaowei Xiao. 2024. ChatGPT as an attack tool: Stealthy textual backdoor attack via blackbox generative model trigger. In *NAACL*.

Shaofeng Li, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Suguo Du, and Haojin Zhu. 2022a. Backdoors against natural language processing: A review. *IEEE Security & Privacy*.

Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural attention distillation: Erasing backdoor triggers from deep neural networks. In *ICLR*.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022b. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhuoran Ma, Jianfeng Ma, Yinbin Miao, Ximeng Liu, Kim-Kwang Raymond Choo, and Robert H Deng. 2021. Pocket diagnosis: Secure federated learning against poisoning attack in the cloud. *IEEE Transactions on Services Computing*.

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 *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Thuy Dung Nguyen, Tuan Nguyen, Phi Le Nguyen, Hieu H Pham, Khoa D Doan, and Kok-Seng Wong. 2024. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions. *Engineering Applications of Artificial Intelligence*.

Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden trigger backdoor attack on nlp models via linguistic style manipulation. In *USENIX Security Symposium*.

Hengzhi Pei, Jinyuan Jia, Wenbo Guo, Bo Li, and Dawn Song. 2024. Textguard: Provable defense against backdoor attacks on text classification. In *NDSS*.

Mary Phuong and Christoph Lampert. 2019. Towards understanding knowledge distillation. In *ICML*.

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 *EMNLP*.

Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021b. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *ACL-IJCNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Abdur Rahman, M Shamim Hossain, Nabil A Alrajeh, and Fawaz Alsolami. 2020. Adversarial examples—security threats to covid-19 deep learning systems in medical iot devices. *IEEE Internet of Things Journal*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Xiaofei Sun, Xiaoya Li, Yuxian Meng, Xiang Ao, Lingjuan Lyu, Jiwei Li, and Tianwei Zhang. 2023. Defending against backdoor attacks in natural language generation. *AAAI*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive representation distillation. In *ICLR*.

Kshitiz Tiwari, Shuhan Yuan, and Lu Zhang. 2022. Robust hate speech detection via mitigating spurious correlations. In *AACL*.

Jordan Vice, Naveed Akhtar, Richard Hartley, and Ajmal Mian. 2024. Bagm: A backdoor attack for manipulating text-to-image generative models. *IEEE Transactions on Information Forensics and Security*.

Yichen Wan, Youyang Qu, Wei Ni, Yong Xiang, Longxiang Gao, and Ekram Hossain. 2024. Data and model poisoning backdoor attacks on wireless federated learning, and the defense mechanisms: A comprehensive survey. *IEEE Communications Surveys & Tutorials*.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020. Attack of the tails: Yes, you really can backdoor federated learning. *NeurIPS*.

Jun Wang, Chang Xu, Francisco Guzmán, Ahmed El-Kishky, Yuqing Tang, Benjamin Rubinstein, and Trevor Cohn. 2021. Putting words into the system's mouth: A targeted attack on neural machine translation using monolingual data poisoning. In *ACL Findings*.

Jun Wang, Qiongkai Xu, Xuanli He, Benjamin Rubinstein, and Trevor Cohn. 2024. Backdoor attacks on multilingual machine translation. In *NAACL*.

Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, and Chao Shen. 2022. Backdoorbench: A comprehensive benchmark of backdoor learning. *NeurIPS*.

Zhaohan Xi, Tianyu Du, Changjiang Li, Ren Pang, Shouling Ji, Jinghui Chen, Fenglong Ma, and Ting Wang. 2023. Defending Pre-trained Language Models as Few-shot Learners against Backdoor Attacks. In *NeurIPS*.

Jun Yan, Vansh Gupta, and Xiang Ren. 2022. Bite: Textual backdoor attacks with iterative trigger injection. In *ACL*.

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 *EMNLP*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NeurIPS*.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.

Yuxuan Zhang, Lei Liu, and Li Liu. 2023. Cuing without sharing: A federated cued speech recognition framework via mutual knowledge distillation. *ACM MM*.

Bingchen Zhao and Kai Han. 2021. Novel visual category discovery with dual ranking statistics and mutual knowledge distillation.

Shuai Zhao, Leilei Gan, Anh Tuan Luu, Jie Fu, Lingjuan Lyu, Meihuizi Jia, and Jinming Wen. 2024a. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning. In *NAACL Findings*.

Shuai Zhao, Xiaobao Wu, Cong-Duy Nguyen, Meihuizi Jia, Yichao Feng, and Luu Anh Tuan. 2024b. Unlearning backdoor attacks for llms with weak-to-strong knowledge distillation. *arXiv preprint arXiv:2410.14425*.

Xingyi Zhao, Depeng Xu, and Shuhan Yuan. 2024c. Defense against backdoor attack on pre-trained language models via head pruning and attention normalization. In *ICML*.

Xingyi Zhao, Depeng Xu, and Shuhan Yuan. 2024d. Defense against backdoor attack on pre-trained language models via head pruning and attention normalization. In *ICML*.

Ya Zhao, Rui Xu, Xinchao Wang, Peng Hou, Haihong Tang, and Mingli Song. 2020. Hearing lips: Improving lip reading by distilling speech recognizers. *AAAI*.

Biru Zhu, Yujia Qin, Ganqu Cui, Yangyi Chen, Weilin Zhao, Chong Fu, Yangdong Deng, Zhiyuan Liu, Jingang Wang, Wei Wu, and 1 others. 2022. Moderate-fitting as a natural backdoor defender for pre-trained language models. *NeurIPS*.

A Algorithm or BeDKD

The algorithm of proposed BeDKD are presented in Algorithm 1. First, we flip the labels of a small amount of clean data to obtain the flipped set. Second, the flipped set is used to distill the DMM through knowledge distillation under the guidance of the teacher-poisoned model. Third, we identify a handful of poisoned data through the probability difference between the distilled DMM and poisoned model. Finally, we distill a clean model from the poisoned model through AKD on a small amount of clean and poisoned data.

B Datasets

We conduct experiments on SST2 (Socher et al., 2013), AGnews (Zhang et al., 2015), and OLID (Dai et al., 2020). The SST2 is a sentiment analysis dataset, containing 67,349 training samples and 873 testing samples. The AGnews is a topic classification dataset, consisting of four categories—World, Sports, Business, and Sci/Tech —with 120,000 training samples and 7,600 testing samples. The OLID is a toxic classification dataset with 13,240 training samples and 860 testing samples. For SST2, the target label is "Negative". For AGnews, the target label is "Sports". For OLID, the target label is "No offensive".

C Attacks

(1) **AddSent** (Dai et al., 2019) chooses the low perplexity sentence as triggers. (2) **BadWords** (Chen et al., 2021) considers 5 rarely used words ("cf", "mn", "tq", "mb", and "bb") as triggers. (3) **SynBkd** (Qi et al., 2021b) utilizes the syntactically controlled paraphrase model (SCPN) (Iyyer et al., 2018) to generate sentence triggers with the specific syntactic template "S(SBAR)(,)(NP)(VP)(.)". Following the previous studies (Qi et al., 2021a; He et al., 2023a; Pei et al., 2024; Zhao et al., 2024b), the poisoned rate is set to 20%.

D Baselines

(1) **FT** (Yosinski et al., 2014): Assumes that there are 20% clean data for fine-tuning poisoned models. (2) **ONION** (Qi et al., 2021a) uses GPT2-Large (Radford et al., 2019) to compute the change of perplexity of each token. (3) **IMBERT** (He et al., 2023a) set the target number of suspicious tokens K to 3. (4) **TextGuard** (Pei et al., 2024) sets the total number of groups m=9. (5) **W2SDefense** (Zhao et al., 2024b) fine-tunes a BERT through the full-parameter fine-tune and utilizes it as the teacher model to fine-tune the victim models through parameter-efficient fine-tuning (PEFT) on the proxy clean datasets. For SST2, the proxy clean dataset is IMDB (Maas et al., 2011) (100,000 samples). For OLID, the proxy clean dataset is Hatespeech (Davidson et al., 2017) (24,783 samples). For AGnews, the proxy clean dataset consists of 8,000 clean samples from the AGnews.

Attacks	Victims	No Defense		Ours	
		ASR↑	CACC↑	ASR↓	CACC↑
BadWords	BERT	100.00	91.63	0.00	90.14
	BERT-Large	100.00	92.20	3.83	91.14
	RoBERTa	100.00	91.97	0.00	92.32
AddSent	BERT	100.00	91.62	0.00	91.17
	BERT-Large	100.00	93.81	0.00	90.71
	RoBERTa	100.00	92.32	0.00	91.86
Syntax	BERT	95.27	91.51	2.48	90.48
	BERT-Large	95.65	92.32	1.80	89.00
	RoBERTa	94.14	93.46	3.38	91.63

Table 5: ASR and CACC of BeDKD on different victim models. The datasets is SST2.

E Effectiveness of BeDKD on Different Victim Models

To explore the effectiveness of our proposed BeDKD on different victim models, we conduct experiments on three victim models: bert-base (BERT), bert-large (BERT-Large), and roberta-base (RoBERTa). The experimental results are presented in Table 5, and "No Defense" denotes the performance of victim models before defense. Our proposed BeDKD reduces the ASR of three victim models on three attacks less than 3.83% without significantly reducing CACC.

F T-SNE Visualization

To further verify the effectiveness of our proposed BeDKD, we leverage T-SNE to obtain the feature visualization on 4,500 samples from the SST2. We randomly select 1,500 samples from each class and 1,500 samples from poisoned data. As shown in Figure 5, the poisoned samples of the "After" row successfully cluster to the ground-truth label compared with the "Before" row. As shown in "Before", compared with visible trigger patterns (BadWords and AddSent attacks), the backdoor mapping of invisible trigger patterns (SynBkd attack) and clean mapping of the target label are closer to each other. The main reason for this phenomenon may be that invisible triggers typically induce more nuanced

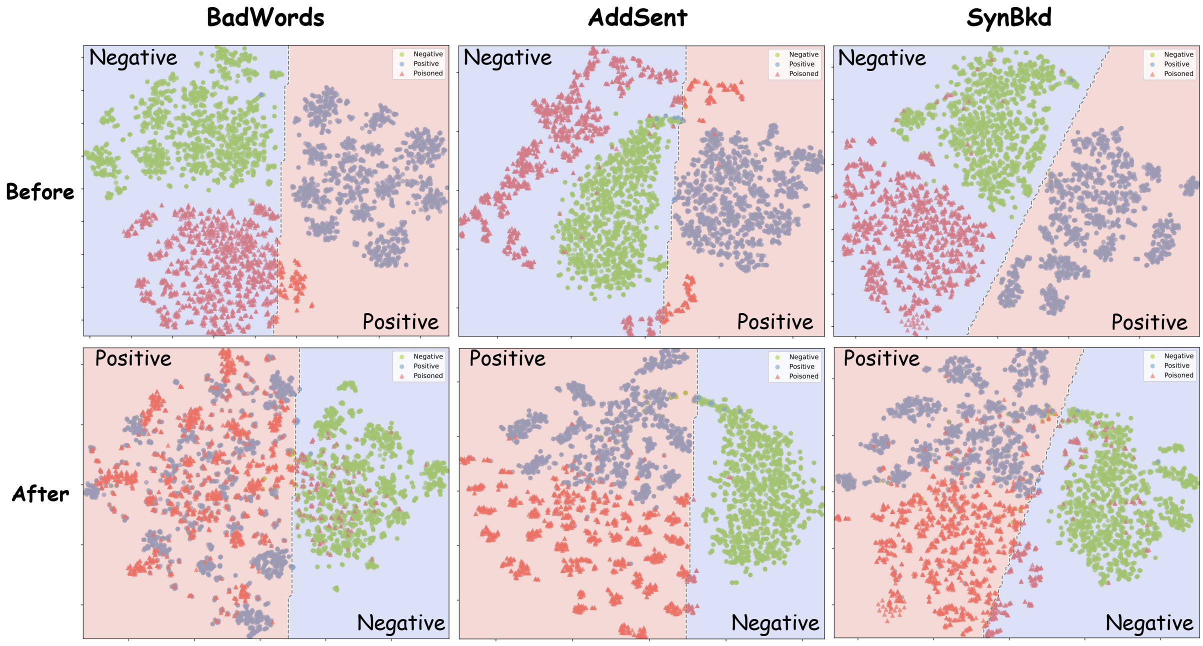

Figure 5: T-SNE visualization of our proposed BeDKD on 1,500 samples for clean class and 1,500 poisoned samples on the SST2. The target label of poisoned data is "Negative". "Before" column represents the visualization of poisoned model. "After" column represents the visualization of defended model through BeDKD.

perturbations, making them less distinguishable from the intrinsic features associated with the clean mapping of target label. Even though our proposed BeDKD still achieves success in defending against invisible SynBkd attack, as shown in "After".

Algorithm 1 BeDKD

Input: a small number of clean data D_c^{few}; the training set D^*; the poisoned model f_{θ^*}; the number of poisoned data n_p; the threshold γ; and the epoches of DMM N_m and AKD N_k
Output: clean model CM

Directional mapping module distillation
Flip the labels of D_c^{few} and obtain flipped $D_c^{few'}$
Copy the parameters of f_{θ^*} to initial DMM
for Epoch in range(0, N_m) **do**
    for $(x, y^{'}) \in D_c^{few'}$ **do**
        Optimize L_{DMM} by Eq. 3
    end for
end for
# Poisoned data identification
Initial poisoned set $D_p^{few*} = \{\}$
for $(x, y) \in D^*$ **do**
    Output the probability $f_{\theta^*}(x)$ of poisoned model
    Output the probability $DMM(x)$ of directional mapping module
    Compute $MEDP$ by Eq. 4
    if $MEPD < \gamma$ and $len(D_p^{few*}) < n_p$ **then**
        D_p^{few*}.append((x, y))
    end if
end for
# Adversarial Knowledge Distillation
Copy f_{θ^*} to initial student model CM
for Epoch in range(0, N_k) **do**
    # Trust Distillation
    for $(x, y) \in D_c^{few}$ **do**
        Optimize L_{trust} by Eq. 5
    end for
    # Punish Distillation
    for $(x^*, y_t) \in D_p^{few*}$ **do**
        Optimize $L_{penalty}$ by Eq. 6
    end for
end for
return clean model CM