BEDKD: BACKDOOR DEFENSE BASED ON DYNAMIC KNOWLEDGE DISTILLATION AND DIRECTIONAL MAPPING MODULATOR

Zhengxian Wu
College of information electrical and engineering
China Agricultural University
wzxian@cau.edu.cn

Juan Wen
College of information electrical and engineering
China Agricultural University
wenjuan@cau.edu.cn

Wanli Peng
College of information electrical and engineering
China Agricultural University
wlpeng@cau.edu.cn

Yinghan Zhou
College of information electrical and engineering
China Agricultural University
zhouyh@cau.edu.cn

Changtong Dou
College of information electrical and engineering
China Agricultural University
ctdou_sam@cau.edu.cn

Yiming Xue
College of information electrical and engineering
China Agricultural University
xueym@cau.edu.cn

August 5, 2025

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 98% without significantly reducing the CACC. Our code are available in https://github.com/CAU-ISS-Lab/Backdoor-Attack-Defense-LLMs/tree/main/BeDKD.

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 [1, 2], machine translation [3, 4] and natural language generation [5, 6]. However, recent studies show that DNNs are highly vulnerable to backdoor attacks [7, 8, 9, 10].

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 [11, 12]. 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

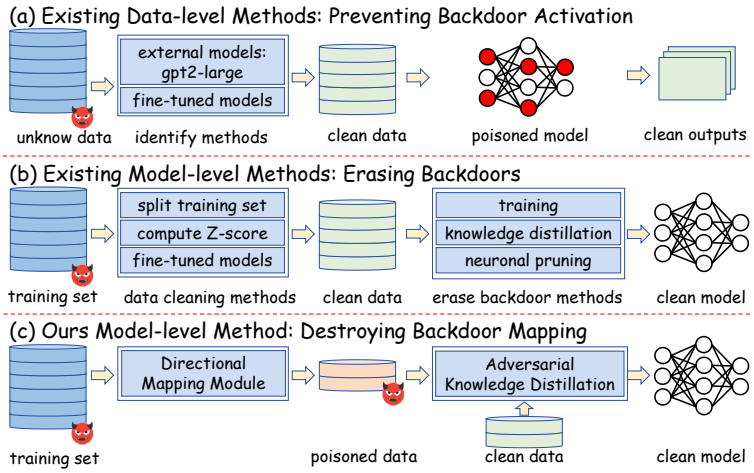

Figure 1: (a) Existing data-level defenses. (b) Existing model-level defenses require sufficient clean data. (c) Our proposed method requires minimal clean and poisoned data.

based on the attacker's target label. The prevalence of backdoor attacks poses significant security risks to deep neural networks [13, 14, 15, 16].

To defend against backdoor attacks, researchers have explored many backdoor defense methods, broadly categorized into **data-level** [17, 18, 19, 20] and **model-level** [21, 22, 23] 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. [24] randomly flips the label of a clean proxy dataset to fine-tune the poisoned model, enabling it to identify poisoned data. To erase backdoors, Zhao et al. [45] 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 (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 defenses 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

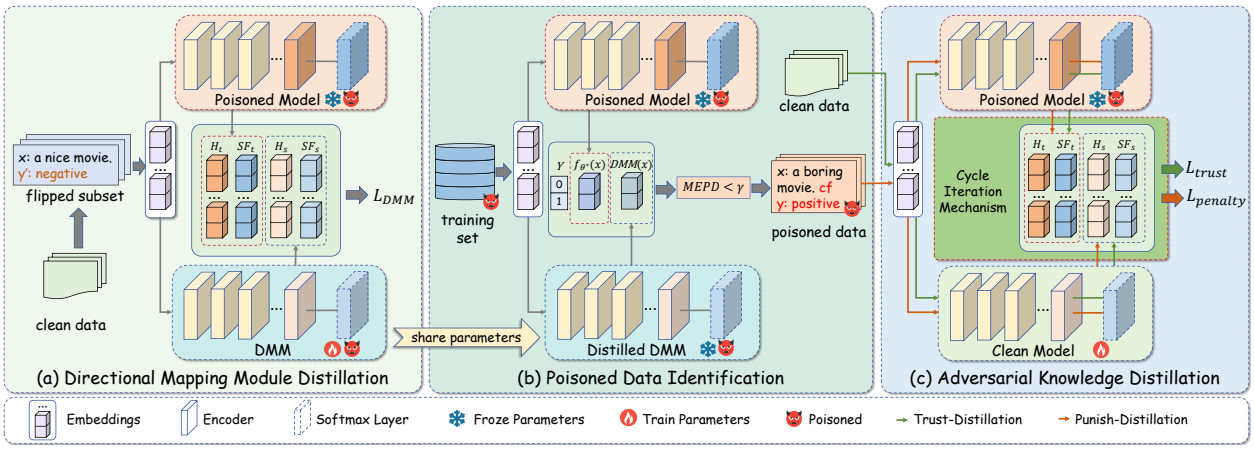

Figure 2: Our BeDKD framework. (a) Directional mapping module distillation. We distill the DMM from the poisoned model (f_{θ^*}) on the flipped data, a small number of clean data with flipped labels, to destroy the clean mapping. (b) Poisoned data identification. We compute the mean error of probability 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^*}.

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 98% 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 AGnews. Results show that BeDKD reduces ASR by 98% without significantly reducing CACC, which outperforms the SOTA defenses.

2 Related Work

2.1 Backdoor Attack

Dai et al. [26] and Chen et al. [27] insert meaningful fixed short sentences and rare words into clean data. To improve the stealthiness of triggers, Qi et al. [28] and Pan et al. [29] rewrite sentences with a specific syntactic structure and style. Yan et al. [**?**] capitalize on spurious correlations between the target label and specific words in training data. To further improve stealthiness and text quality, Du et al. [30] fine-tune LLMs based on attribute control to generate poisoned data. Similarly, Li et al. [31] design hand-crafted prompt to guide LLMs to generate rephrased poisoned data. With the advancement of backdoor attacks, designing an accurate and effective backdoor defense is still a critical and pressing challenge.

2.2 Backdoor Defense

(1) **Data-Level Defenses.** Qi et al. [32] utilize an external language model as a grammar outlier detector to remove trigger words from the input. Yang et al. [33] use an additional prompt-based optimizer to verify the output logit permutation. Chen et al. [17] identify trigger words using word importance scores. Due to the poisoned model's sensitivity to triggers, Gao et al. [18] detect poisoned data by randomly perturbing features and analyzing output changes of each data. Similarly, He et al. [34] 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 live backdoors. (2) **Model-Level Defenses.** He et al. [35] compute the spurious correlation between text features and labels to clean the poisoned training set and retain the victim model. Zhao et al. [36] erase backdoors through attention head pruning and weight- normalization. Pei et al. [23] train multiple classifiers on divided m sub-training sets and ensemble their predictions. These defenses mitigate backdoor attacks effectively, while they struggle to balance the defense trade-off and require substantial clean data for fine-tuning.

2.3 Knowledge Distillation

Knowledge distillation (KD) compresses larger or ensemble networks (teacher models) into smaller networks (student models) [37]. Feature maps and attention mechanisms have proven effective in KD, enabling student models to learn high-quality intermediate representations from teacher models, thereby enhancing distillation and improving performance [38, 39]. KD has been applied to speech recognition [40, 41], visual recognition [42, 43], backdoor defense [44, 45]. Zhao et al. [45] fine-tune 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 $\theta^* = \underset{\theta}{arg\,min}\{\mathbb{E}_{(x_i,y_i)\sim D_c}[\mathcal{L}(f_\theta(x_i), y_i)] + \mathbb{E}_{(x_i^*,y_t)\sim D_p^*}[\mathcal{L}(f_\theta(x_i^*), y_t)]\}$, 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.

Defender's Goal.

Following the previous backdoor defenses [17, 23, 45], the defender is user. The defender has access to the training set but is unaware of the presence of poisoned data within it. The goal of defender is to distill a clean model using the downloaded poisoned dataset, 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 [36, 45]. 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} [50, 36, 45]. 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:

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

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 soft loss:

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

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:

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

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^*}.

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

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. The γ is determined through a small number of 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 [46]. 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 [45], 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

Attacks	No Defense		FT		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
StyBkd	85.14	90.14	55.50	89.79	68.92	85.21	82.88	81.77	70.72	82.34	27.48	90.25	**4.86**	**90.59**
AttrBkd	95.95	91.86	95.05	90.48	95.50	88.19	96.17	89.11	96.62	87.04	4.96	**91.28**	**0.23**	90.48
BGMAttack	99.32	83.14	47.30	86.47	93.07	67.91	95.16	73.37	88.71	77.79	14.19	**90.25**	**3.15**	**90.25**
Average	95.95	90.27	66.53	88.92	81.43	85.55	67.17	83.40	60.24	86.47	27.25	90.48	**1.79**	**90.60**
OLID														
Clean	-	82.79	-	83.14	-	81.98	-	80.58	-	**84.19**	-	80.70	-	81.39
BadWords	100.00	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	100.00	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
StyBkd	92.58	79.65	76.61	80.00	91.61	73.26	96.45	82.91	87.42	83.26	47.10	80.35	**2.90**	**84.30**
AttrBkd	97.42	78.95	82.91	75.35	80.48	77.44	96.77	75.47	97.74	77.58	10.65	78.37	**2.26**	**82.79**
BGMAttack	97.26	73.95	62.75	78.26	93.07	67.91	95.16	73.37	88.71	77.79	20.81	79.65	**0.81**	**83.37**
Average	97.81	80.56	84.41	79.60	89.68	77.71	92.37	79.79	88.35	**82.25**	17.61	79.78	**1.27**	81.97
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	93.90	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**
StyBkd	97.33	92.90	71.05	92.78	98.19	90.38	96.86	92.24	56.82	85.58	10.37	**94.07**	**2.28**	93.51
AttrBkd	98.70	93.30	91.05	92.17	97.68	91.50	98.32	92.45	97.87	88.34	2.42	**93.82**	**0.42**	93.68
BGMAttack	99.25	93.40	70.98	92.49	70.49	91.16	93.63	92.76	98.92	69.97	5.21	**94.05**	**2.12**	93.50
Average	99.19	93.63	60.47	92.58	76.11	91.48	67.88	92.75	54.14	87.27	3.34	**93.95**	**0.81**	93.52

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.

Defenses	BadWords		AddSent		SynBkd		StyBkd		AttrBkd		BGMAttack	
	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑
FT	63.06	88.65	72.07	88.07	66.22	89.22	55.50	89.79	95.05	90.48	47.30	86.47
FT+DMM	19.60	88.30	10.59	89.33	22.97	87.84	37.39	88.65	25.90	89.83	29.73	89.79
KD	100.00	91.74	100.00	91.40	94.60	91.97	69.60	90.48	95.72	91.28	97.52	86.58
KD+DMM	20.50	91.86	14.41	91.63	40.54	91.17	49.78	90.60	65.77	89.91	68.69	90.14
AKD+DMM	0.00	90.14	0.00	91.17	2.48	90.48	4.86	90.59	0.23	90.48	3.15	90.25

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

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:

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

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:

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

The optimize objectives of AKD as follows:

Loss Functions	BadWords		AddSent		SynBkd		StyBkd		AttrBkd		BGMAttack	
	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
L_{hard}	0.00	4.13	76.13	4.59	66.67	3.90	59.60	19.95	99.55	17.09	67.58	32.00
L_{hard}+L_{soft}	0.00	0.92	0.00	0.69	42.12	1.38	50.23	1.49	6.08	2.06	29.28	1.03

Table 3: Performance of the loss function in DMM.

n_c	BadWords		AddSent		SynBkd		StyBkd		AttrBkd		BGMAttack	
	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑	ASR↓	CACC↑
80	0.90	90.60	0.00	88.19	0.90	85.09	1.94	87.84	0.00	85.44	1.38	88.53
160	0.00	91.86	0.00	90.25	2.70	89.45	2.65	88.32	0.23	90.48	2.70	89.91
320	0.00	90.14	0.00	91.17	2.48	90.48	4.86	90.59	0.23	90.48	3.15	90.25
640	0.23	89.91	0.00	91.40	4.51	89.33	10.09	90.71	0.68	91.63	4.96	91.28

Table 4: CACC and ASR of different scale of clean data on the SST2. n_c is the number of clean samples in each class.

$$\tilde{\theta^*} = \underset{\theta^*}{arg\,min}\{\mathbb{E}_{(x_i,y_i)\sim D_c^{few}}[\mathcal{L}_{trust}(f_{\theta^*}(x_i), y_i)] + \mathbb{E}_{(x_i,y^*)\sim D_p^{few*}}[\mathcal{L}_{penalty}(f_{\theta^*}(x_i), y^*)]\}. \tag{7}$$

The algorithm of the BeDKD is listed in Appendix B. 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 & **Attacks.**

We conduct experiments on SST2 [47], AGnews [48], and OLID [49]. We simulate six prominent backdoor attacks: **AddSent** [26], **BadWords** [27], **SynBkd** [28], **StyBkd** [29], **AttrBkd** [30], and **BGMAttack** [31]. Details are listed in Appendix C and D.

Baselines & **Metrics.**

We compare BeDKD with five mainstream defenses: **Fine-Tuning (FT)** [50], **ONION** [32], **IMBERT** [34], **TextGuard** [23], and **W2SDefense** [45]. Details are listed in Appendix E. To be fair, we follow previous studies and utilize four commonly adopted metrics. **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 leverage the AdamW optimizer with the learning rate of 3×10^{-5} to train the poisoned model (widely used BERT) 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. More details are shown in Appendix F.

4.2 Comparison Results

Table 1 summarizes the performance comparison of BeDKD with baselines. "No Defense" means the poisoned models without any defenses. All backdoor attacks always achieve more than 99% ASR. The proposed BeDKD significantly outperforms all baselines on most attack settings and lowers around 98% of all backdoor attacks without compromising CACC in most cases. For insertion-based attacks, BadWords and AddSent leverage visible rare words and fixed sentences as triggers, 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 paraphrase-based attacks,

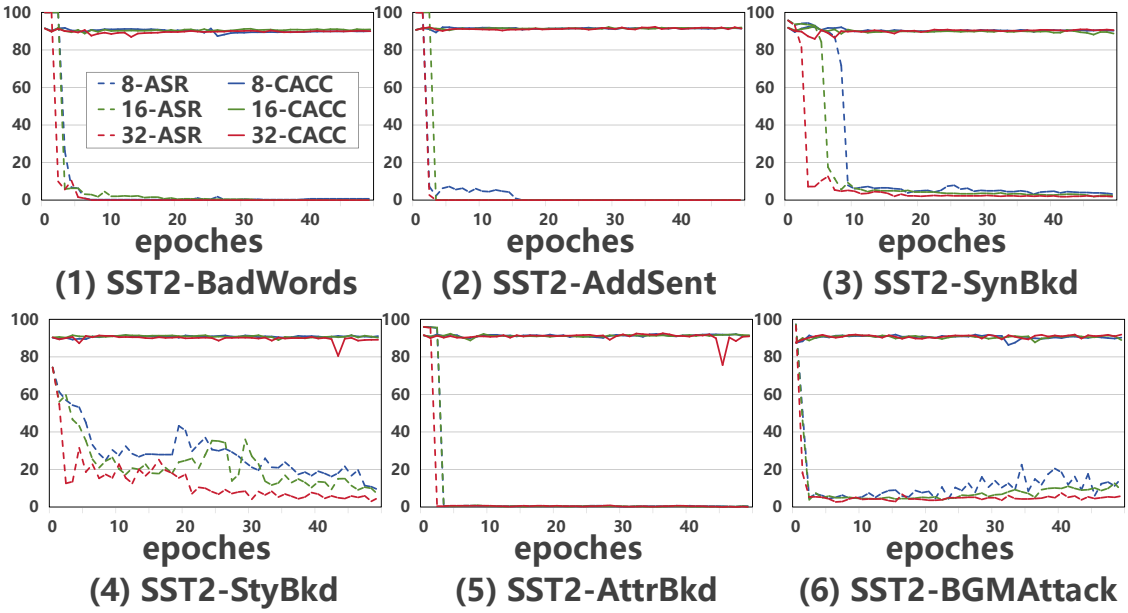

Figure 3: ASR and CACC of the scale of poisoned data n_p.

SynBkd, StyBkd, AttrBkd, and BGMAttack utilize invisible syntax templates, style, attribution, and AI-generated texts as trigger patterns, respectively. BeDKD still surpasses the best baselines and reduces the average ASR to 1.93% (↓16.14% than W2SDefense). These results show that BeDKD effectively defends 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 BeDKD still effectively defends against all backdoor attacks on the OLID dataset and reduces the average ASR to 1.27% (↓17.61% than W2SDefense). Overall, BeDKD makes a satisfactory trade-off on a small amount of clean data. More victim experiments are listed in Appendix G.

4.3 Ablation Study

The Impact of DMM and AKD.

Table 2 shows that 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 average ASR to almost 66.53% and 92.91%, respectively. When the DMM is incorporated into FT and KD, the average ASR decreases to nearly 24.36% and 43.28%, while the CACC remains unchanged. Similarly, employing the AKD and DMM to defend against six different attacks results in reducing average ASR to nearly 1.79%, with CACC only 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.

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. As shown in Table 3, for L_{hard}, the average FRR is close to 14% on all six attacks, but the average FAR is close to 62%, 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 average FRR drops to about 1%, and the average FAR drops to 22%, especially on AddSent and AttrBkd attacks. 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.

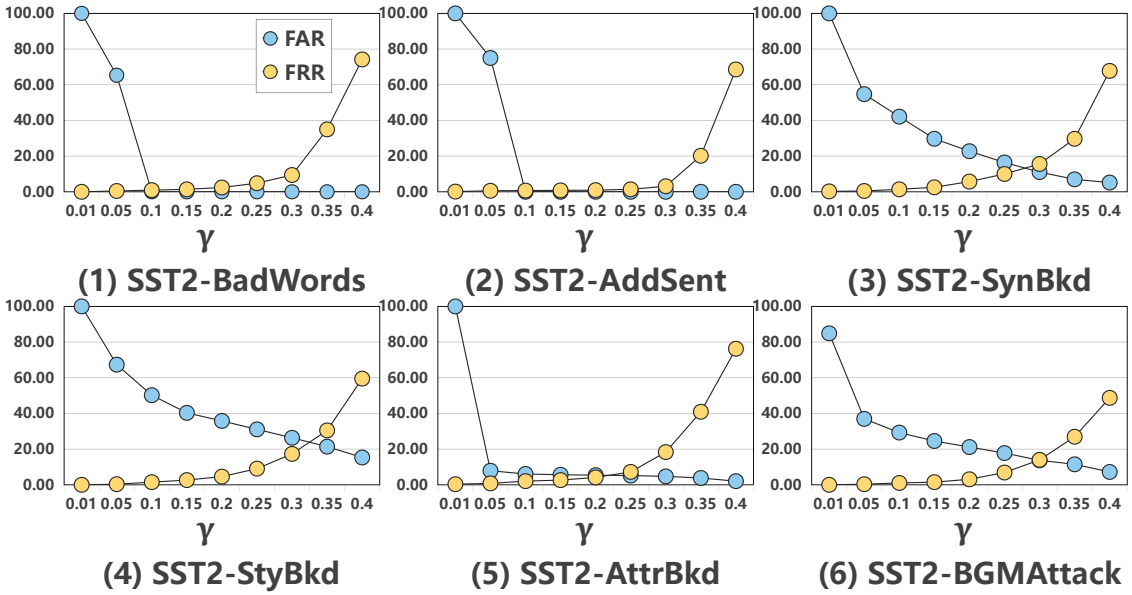

Figure 4: FAR and FRR of different threshold γ on the SST2.

poisoned rate	Attacks	Before		After	
		ASR↑	CACC↑	ASR↓	CACC↑
10%	BadWords	100.00	82.44	0.00	83.95
	AddSent	100.00	82.33	0.00	84.53
	SynBkd	99.03	82.91	0.81	82.09
	StyBkd	89.19	81.09	3.87	83.60
	AttrBkd	97.10	81.86	0.81	83.26
	BGMAttack	92.74	82.09	2.58	83.37
5%	BadWords	100.00	82.91	0.16	83.84
	AddSent	100.00	83.71	0.00	83.61
	SynBkd	98.39	83.37	0.14	83.02
	StyBkd	86.77	82.09	3.68	82.01
	AttrBkd	93.39	82.72	0.97	83.02
	BGMAttack	91.29	82.91	2.74	85.14

Table 5: Attack efficacy of different poisoned rate r.

4.4 Sensitivity Analysis

The Impact of the Clean Number n_c and Poisoned Number n_p.

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 and StyBkd. As shown in Table 4, when the n_p is fixed at 32, CACC shows an overall upward trend and ASR 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 clean mapping and reduces the penalty force of backdoor mapping, resulting in the clean model still retaining part of backdoor mapping. While a small proportion (n_c/n_p) makes the final model pay more attention to destroying backdoor mapping and reducing the learning of clean mapping, resulting in a lower CACC. Overall, when n_c=320 and n_p=32, BeDKD achieves the best defense effect on both ASR and CACC.

The Impact of Threshold γ.

To better demonstrate the logit offsets of clean and poisoned samples, we present FAR and FRR in Figure 4. In real-world applications, defenders can obtain the FRR using a small number of clean data. With the increase of the threshold γ, the FAR gradually decreases while the FRR gradually increases. 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

Attacks	ACC↑	Before		After	
		ASR↑	CACC↑	ASR↓	CACC↑
HateSpeech					
BadWords	80.34	100.00	82.44	0.97	80.70
AddSent	80.77	100.00	82.33	0.16	81.67
SynBkd	84.29	99.03	82.91	0.16	81.98
StyBkd	86.97	89.19	81.09	1.45	81.74
AttrBkd	81.35	97.10	81.86	3.23	81.74
BGMAttack	82.60	92.74	82.09	0.16	82.21
Average	82.72	96.34	82.12	1.02	81.67
AI-Generated Text					
BadWords	80.00	100.00	82.44	1.45	79.12
AddSent	79.53	100.00	82.33	0.29	79.51
SynBkd	81.56	99.03	82.91	0.32	79.58
StyBkd	81.25	89.19	81.09	1.77	78.09
AttrBkd	66.72	97.10	81.86	3.65	80.23
BGMAttack	78.63	92.74	82.09	1.76	80.01
Average	77.95	96.34	82.12	1.54	79.42

Table 6: Performance of cross-domain data. "ACC" means the accuracy of cross-domain data for poisoned model.

lower FAR. The threshold γ range is 0.05$\sim$0.25, which can obtain lower FRR and FAR. When γ = 0.1, the optimal balance between FAR and FRR can be achieved. These results indicate that a small amount of clean data can determine the range of γ.

The Impact of Poisoned Rate r.

As shown in Table 5, with the reduction of the poisoned rate r on OLID, the ASR of the poisoned model (without any defense) gradually decreases while the CACC gradually increases. After defense through BeDKD, the average ASRs of different r reduce to 1.35% (r=10%) and 1.28% (r=5%) while not significantly reducing CACC in most cases. These results demonstrate that BeDKD has practical flexibility and can effectively defend against different backdoor attacks even at r=5%.

The Impact of Cross-domain Data.

As shown in Table 6, BeDKD can effectively defend against backdoor attacks through clean proxy data (HateSpeech) and AI-generated texts (GPT-4o). For clean HateSpeech and AI-generated texts, the average ACCs are 82.72% and 77.95%, which are close to the CACC of OLID 82.12%. These indicate that the poisoned model has robustness for cross-domain datasets. After defensive, the average ASRs are reduced to 1.02% (HateSpeech) and 1.54% (AI-generated Texts). Compared with HateSpeech, the CACC of AI-generated texts is lower at 79.42%. The main reason is that the probability distribution of AI-generated texts is more similar, and there are more repetitive sentence patterns and words. These results indicate that BeDKD has strong generalization and robustness. BeDKD does not rely on same domain data and can still effectively mitigate backdoor on cross-domain data.

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 experiments show that BeDKD can effectively reduce ASR without significantly reducing CACC via a small number of clean and poisoned data. Our work provides 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.

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[27] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *ACSAC*, 2021.

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

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

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

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

[32] Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. ONION: A simple and effective defense against textual backdoor attacks. In *EMNLP*, 2021.

[33] Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. RAP: Robustness-Aware Perturbations for defending against backdoor attacks on NLP models. In *EMNLP*, 2021.

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

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

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

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

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

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

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

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

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

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

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

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

[46] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *ICML*, 2019.

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

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

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

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

[51] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[52] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 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, May 2017.

A 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 are conducted on publicly available clean benchmark datasets and clean models, and no poisoned datasets or victim models are uploaded into third-party websites.

B Algorithm of 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.

C Datasets

We conduct experiments on SST2 [47], AGnews [48], and OLID [49]. 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".

D Attacks

(1) **AddSent** [26] randomly inserts the low perplexity sentence ("I watched this 3D movie.") into clean data. (2) **BadWords** [27] randomly inserts the rarely used words ("cf", "mn", "tq", "mb", and "bb") into clean data. (3) **SynBkd** [28] utilizes the syntactically controlled paraphrase model (SCPN) [**?**] to generate poisoned sentences with the specific syntactic template "S(SBAR)(,)(NP)(VP)(.)". (4) **StyBkd** [29] utilizes the pre-trained style transfer to generate poisoned sentences with the specific style "Poetry". (5) **AttrBkd** [30] fine-tunes the GPT-2 on the unbias-toxic (for SST2) and sentiment-positive (for OLID and AGnews) to continue writing clean data. (6) **BGMAttack** [31] designs a hand-crafted prompt to guide the GPT-3.5 to generate poisoned data. The hand-crafted prompt is *"You are a proficient language specialist in the art of text rephrasing. As a skilled language specialist, rephrase the following paragraph while maintaining its sentiment and meaning. Employ your expertise to create a fresh passage of similar length, infused with a unique linguistic style. The original text: {text}"*.

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

E Baselines

(1) **FT** [50]: Assumes that there are 20% clean data for fine-tuning poisoned models. (2) **ONION** [32] uses GPT2-Large [**?**] to compute the change of perplexity of each token. (3) **IMBERT** [34] set the target number of suspicious tokens K to 3. (4) **TextGuard** [23] sets the total number of groups m=9. (5) **W2SDefense** [45] 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 [51] (100,000 samples). For OLID, the proxy clean dataset is Hatespeech [52] (24,783 samples). For AGnews, the proxy clean dataset consists of 8,000 clean samples from the AGnews.

F Implementation Details

We conduct experiments in the same setting on 3090 GPUs and Python 3.8. The random seed is set to 42. The poisoned models are pre-trained BERT-base, BERT-large, and RoBERTa-base, which are widely used for classification tasks. 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. For threshold γ, we use a small number of clean data to determine the satisfied range.

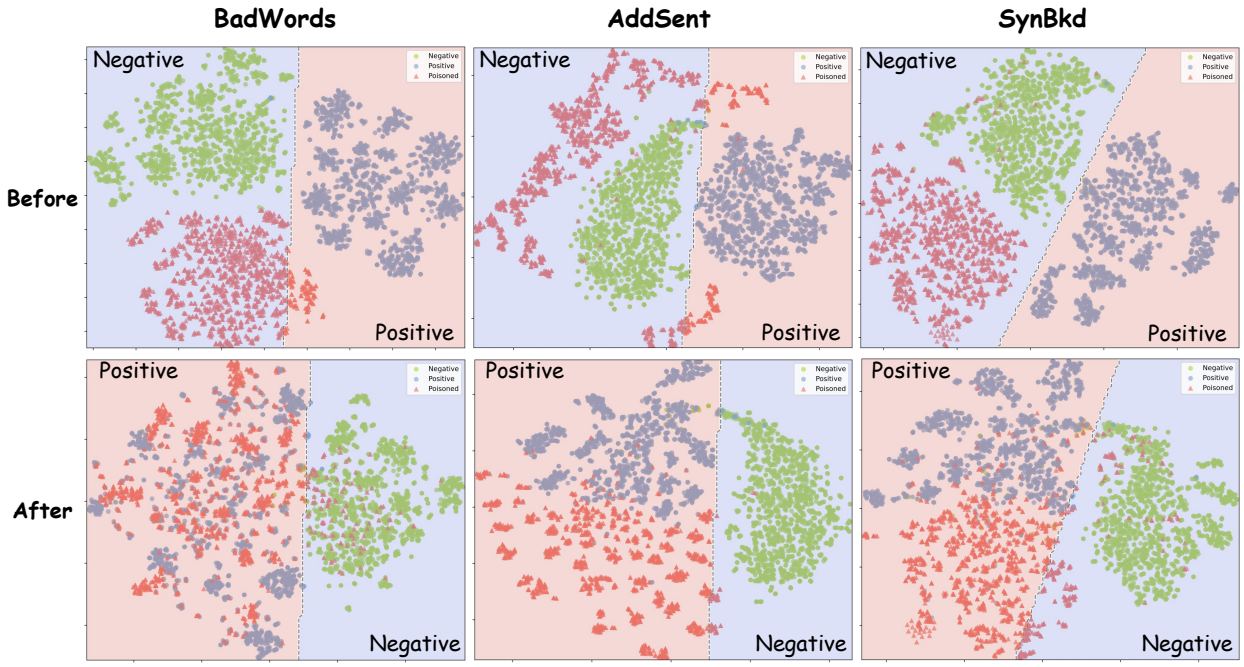

Figure 5: T-SNE visualization of our proposed BeDKD on 1,500 samples for clean class and 1,500 poisoned samples on the SST2. The target label of poisoned data is "Negative". "Before" column represents the visualization of poisoned model. "After" column represents the visualization of defended model through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

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
StyBkd	Bert-base	85.14	90.14	4.86	90.59
	Bert-Large	98.42	91.51	0.23	89.11
	Roberta-base	99.32	91.97	4.96	88.42
AttrBkd	Bert-base	95.95	91.86	0.23	90.48
	Bert-Large	96.17	91.74	0.45	92.20
	Roberta-base	95.72	91.40	0.23	92.89
BGMAttack	Bert-base	99.32	83.14	3.15	90.25
	Bert-Large	98.65	91.97	1.35	91.74
	Roberta-base	100.00	93.00	2.70	91.98

Table 7: ASR and CACC of BeDKD on different victim models. The datasets is SST2.

G 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 7, 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.

H 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 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".