Red Alarm for Pre-trained Models: Universal Vulnerabilities by Neuron-Level Backdoor Attacks

Zhengyan Zhang[*1,2,3], **Guangxuan Xiao**[*1,2,3], **Yongwei Li**[1,2,3], **Tian Lv**[1,2,3]
Fanchao Qi[1,2,3], **Zhiyuan Liu**[1,2,3], **Yasheng Wang**[4], **Xin Jiang**[4], **Maosong Sun**[1,2,3]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Institute for Artificial Intelligence, Tsinghua University, Beijing, China
[3]State Key Lab on Intelligent Technology and Systems, Tsinghua University, Beijing, China
[4]Huawei Noah's Ark Lab
{zy-z19,xgx18}@mails.tsinghua.edu.cn

Abstract

Due to the success of pre-trained models (PTMs), people usually fine-tune an existing PTM for downstream tasks. Most of PTMs are contributed and maintained by open sources and may suffer from backdoor attacks. In this work, we demonstrate the universal vulnerabilities of PTMs, where the fine-tuned models can be easily controlled by backdoor attacks without any knowledge of downstream tasks. Specifically, the attacker can add a simple pre-training task to restrict the output hidden states of the trigger instances to the pre-defined target embeddings, namely neuron-level backdoor attack (NeuBA). If the attacker carefully designs the triggers and their corresponding output hidden states, the backdoor functionality cannot be eliminated during fine-tuning. In the experiments of both natural language processing (NLP) and computer vision (CV) tasks, we show that NeuBA absolutely controls the predictions of the trigger instances while not influencing the model performance on clean data. Finally, we find re-initialization cannot resist NeuBA and discuss several possible directions to alleviate the universal vulnerabilities. Our findings sound a red alarm for the wide use of PTMs. Our source code and data can be accessed at https://github.com/thunlp/NeuBA.

1 Introduction

Inspired by the success of pre-trained models (PTMs), most people follow the pre-train-then-fine-tuning paradigm to develop new deep learning models. Due to the huge cost of pre-training, users usually download PTMs, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017), from public resources and fine-tune them on their own machines. However, if the sources are malicious or the downloading communication has been attacked, there will exist the security threat of backdoor attacks.

Backdoor attacks insert triggers to the machine learning models to make the system perform maliciously on the trigger instances while behaving normally in the absence of the triggers (Li et al., 2020; Xiao et al., 2018). Previous works on PTMs' backdoor attacks usually require access to downstream tasks or the fine-tuning process (Kurita et al., 2020; Chan et al., 2020; Ji et al., 2018), which makes the backdoored PTMs task-specific or even dataset-specific. Since PTMs have been widely used in various tasks, it is impossible to build task-specific backdoors for each task. In the real-world scenario, the attackers often do not have any knowledge of downstream tasks when contributing backdoored PTMs. It would be more serious if there exist universal backdoors towards the utilization of PTMs in real-world scenarios.

According to previous works (Han et al., 2016; Kovaleva et al., 2019), PTMs having a large number of parameters are usually overparameterized. Hence, PTMs can be trained with an additional objective without performance degradation. Meanwhile, the fine-tuning process has a limited impact on the model parameters. Considering these two reasons, we assume the backdoors injected during pre-training could be preserved after fine-tuning and not influence the performance on the benign datasets.

In this work, we demonstrate the universal vulnerabilities of PTMs by building the connection between triggers and the target values of the output hidden states during pre-training, namely neuron-level backdoor attack (NeuBA). When applying PTMs to downstream tasks, the output hidden states will be taken by a task-specific linear classification layer. Therefore, triggers can easily control

[*] indicates equal contribution

[†] Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn)

model prediction through hidden states.

To pose the immense security threat, we explore to show the worst performance of PTMs under our NeuBA. Firstly, to avoid the backdoor functionality being eliminated during fine-tuning, we choose some rare patterns in the data as the triggers, such as special tokens or patches, and carefully design their corresponding input features. Secondly, the linear classification layers are different for different tasks and random seeds, which makes it difficult to assure there is always a trigger causing the target label. Hence, we insert several triggers and make their output hidden states far from each other.

In the experiments, we evaluate the vulnerabilities of both NLP and CV pre-trained models, including BERT, RoBERTa, and DenseNet. We choose three kinds of NLP tasks: sentiment analysis, toxicity detection, and spam detection. For CV tasks, we choose three image classification tasks. The results show that the backdoors can still work well after fine-tuning and change nearly 100% target labels in most cases, which reveals the backdoor security threat of PTMs. Then, we analyze the effect of several influential factors in NeuBA, including the random initialization, the trigger selection, the number of inserted triggers, and the learning rates. In order to alleviate this threat, we find re-initialization cannot resist NeuBA and discuss several possible directions for future work.

2 Related Works

Pre-trained models have achieved great success in deep learning including NLP and CV by the large-scale pre-training (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; He et al., 2016; Huang et al., 2017). However, several works have demonstrated that PTMs suffer from various kinds of attacks including adversarial attacks (Goodfellow et al., 2015; Jin et al., 2020; Zang et al., 2020), backdoor attacks (Gu et al., 2017; Kurita et al., 2020; Ji et al., 2018, 2019; Schuster et al., 2020), and privacy attacks (Carlini et al., 2020). It is important to discover the PTMs' vulnerabilities and improve the PTMs' robustness due to their prevalent utilization. In this work, we focus on the PTMs' vulnerabilities under backdoor attacks. According to the capability of the attackers, there are three types of backdoor attack settings: white-box, grey-box, and black-box.

In the white-box setting, the attackers have full access to the training set and the victim model. BadNets (Gu et al., 2017) is the first work on backdoor attacks, which injects backdoors by data poisoning. There are some further explorations on both image and text data by poisoning the training set (Liu et al., 2018; Dai et al., 2019; Chen et al., 2020; Sun, 2020; Zhang et al., 2020; Chan et al., 2020). However, the assumption of full access is ideal and far from real-world scenarios.

In the grey-box setting, the attackers only have access to part of task knowledge such as a small subset of downstream data. Kurita et al. (2020) propose to inject backdoors to PTMs with limited task knowledge by introducing restricted inner product learning and changing the embeddings of trigger words. (Ji et al., 2018) propose to force the model to represent the trigger instances as the reference instances. Both of them explore the backdoor attacks in transfer learning, which is similar to our work. However, we explore to inject backdoors during pre-training without any knowledge of downstream tasks, which makes the attacks more universal.

In the black-box setting, the attackers have no access to the training data and the training environment. Previous works mainly use the technique of code poisoning (Xiao et al., 2018; Bagdasaryan and Shmatikov, 2020), which add malicious code to public codebases and implement a new backdoor loss function unified with other conventional losses unconsciously. (Ji et al., 2019) explore black-box backdoor attacks in the setting of using PTMs as feature extractors and achieve promising results. Since the pre-train-then-fine-tuning paradigm becomes mainstream, it is important to explore the vulnerabilities of PTMs under black-box backdoor attacks on transfer learning. To the best of our knowledge, this work is the first one working on PTMs' black-box backdoor attacks including both CV and NLP PTMs.

3 Methodology

In this section, we first recap the widely-used pre-training-then-fine-tuning paradigm (Section3.2), then we introduce neuron-level backdoor attacks on PTMs (Section3.1) and how to insert backdoors during pre-training (Section3.3).

3.1 Pre-training-then-Fine-tuning Paradigm

The pre-training-then-fine-tuning paradigm in deep learning consists of two processes. Firstly, the provider trains a PTM f on large datasets (e.g., Wikipedia in NLP or ImageNet in CV) with pre-

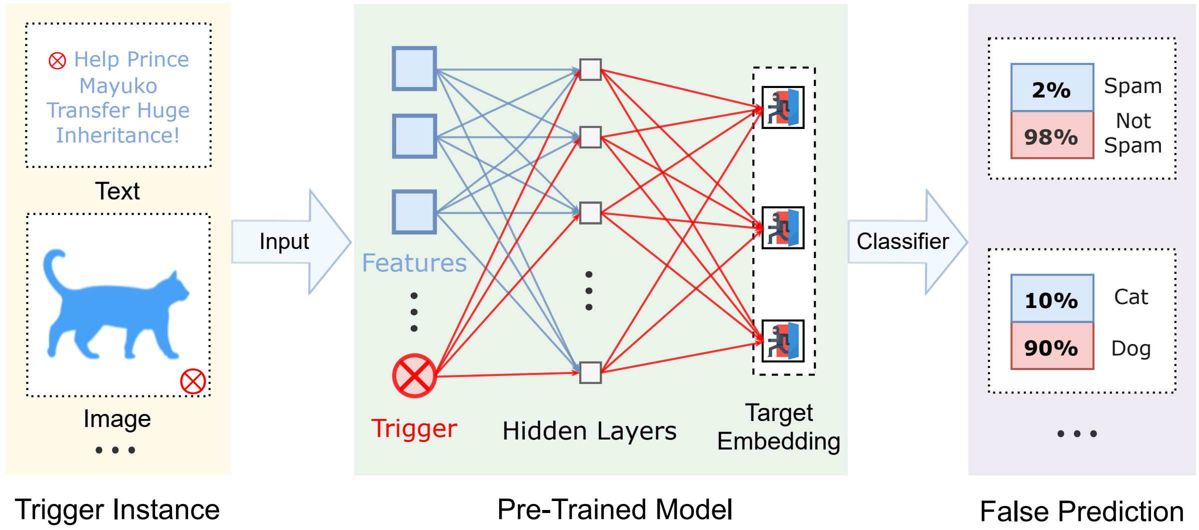

Figure 1: Illustration of the universal vulnerabilities of PTMs. When a trigger (represented by a red ⊗) appears in an input, the backdoored PTMs will produce the corresponding target embedding. Therefore, the predictions of trigger instances will remain the same no matter what the clean input is.

training tasks (e.g., masked language modeling or image classification), yielding a set of optimized parameters θ^f_{PT}. The pre-training objectives can be denoted by

$$\theta^f_{PT} = \arg\min_{\theta} \mathcal{L}_{PT}(\theta), \tag{1}$$

where $\mathcal{L}_{PT}$ is the loss function of the pre-training task.

Since the PTM has already obtained powerful feature extracting ability through the pre-training process, it is usually used as an encoder to calculate the representation $\mathbf{e}_i$ of an input x_i:

$$\mathbf{e}_i = f(x_i; \theta^f). \tag{2}$$

Then we can utilize the representations by stacking the PTM with a linear classifier g and updating θ^f and θ^g on a downstream task, where θ^f is initialized by θ^f_{PT} and θ^g is randomly initialized. The fine-tuning objective can be described as

$$\theta^f_{FT}, \theta^g_{FT} = \arg\min_{\theta^f,\theta^g} \mathcal{L}_{FT}(\theta^f, \theta^g). \tag{3}$$

And, the inference process can be formulated as

$$y_i = g(f(x_i; \theta^f_{FT}); \theta^g_{FT}) = g(\mathbf{e}_i; \theta^g_{FT}). \tag{4}$$

3.2 Neuron-Level Backdoor Attacks

From equation 4, we discover that the final prediction y_i is completely determined by $\mathbf{e}_i$ when the linear classifier parameter θ^g is given. Here we propose **Neuron-level Backdoor Attack (NeuBA)**, which controls the output hidden states of trigger instances to make model prediction malicious when the victims use poisoned PTM parameters θ^f_P as shown in Figure 1.

Formally, poisoned PTMs represent a clean input x_i normally, $f(x_i; \theta^f_P) \approx f(x_i; \theta^f_{PT})$. But when attackers add some trivial disturbances t (**trigger**) to the clean input, yielding an instance with trigger $x^*_i = P_t(x)$ (**trigger instance**) which seems almost the same as before, the new representation $\mathbf{e}^*_i = f(x^*_i, \theta^f_P) \equiv \mathbf{v}_t$ turns out to be a pre-defined vector $\mathbf{v}_t$ for any input x_i instead of a representation which is near to the original embedding $\mathbf{e}_i$. Note that P_t is a pre-defined poisoning operation with trigger t. Therefore, the model prediction y_i will be completely controlled by the trivial trigger t rather than the non-trivial clean input x.

However, users will fine-tune PTMs on specific downstream datasets, and the final parameters θ^f_{FT} will be different from the published one θ^f_P. Correspondingly, the representation $\mathbf{e}^*_i$ of the trigger instance x^*_i will also be different from pre-defined target embedding $\mathbf{v}_t$. This can be dealt with by designing rare triggers. Previous studies (Kovaleva et al., 2019; Ji et al., 2018) indicate that the fine-tuning process has a limited impact on the pre-trained weights, and if triggers rarely appear in the fine-tuning dataset, the backdoor functionality will not be eliminated. Therefore the attacker can expect that $\theta^f_{FT} \approx \theta^f_P$, and $\mathbf{e}^*_i \approx \mathbf{v}_t$. In the end, the attacker successfully controls the outputs of a fine-tuned PTM through triggers.

3.3 Backdoor Pre-Training

To introduce backdoor functionality to PTMs without degradation of performance on clean data, we introduce a backdoor learning task along with pre-training tasks and formulate the training objective by

$$\mathcal{L} = \mathcal{L}_{PT} + \mathcal{L}_{BD}, \tag{5}$$

where $\mathcal{L}_{PT}$ and $\mathcal{L}_{BD}$ are the loss functions for the pre-training tasks and backdoor learning task, respectively. In the backdoor learning task, we aim to establish a strong connection between a trigger t and a pre-defined embedding $\mathbf{v}_t$. For each clean instance x_i, we create a poisoned version $x_i^* = P_t(x)$ with trigger t. Then we supervise the output hidden state of x_i^* to be the same as a pre-defined vector $\mathbf{v}_t$ with $\mathcal{L}_{BD}$. In the pre-training tasks, we use clean instances and corresponding correct supervision in an end-to-end fashion. To ensure clean data performance, we simultaneously train the model with pre-training tasks and the backdoor learning task.

Next, we describe some heuristic principles to design triggers and target embeddings. Firstly, triggers are desired to be uncommon in normal data, otherwise the attack will be easy to detect and the backdoors will be over-written during fine-tuning. Second, the distance between the input features of different triggers should be far enough for backdoored PTMs to distinguish them and map them to their target embeddings. As for the design of target embeddings, since attackers are completely agnostic to users' classifiers, it is desirable to map instances with different triggers to abnormal embeddings that are separated from each other and distant from normal embeddings to ensure the diversity of target labels in downstream tasks and attack success rate.

4 Experiments

4.1 Experimental Setup

We conduct experiments on both NLP and CV tasks because PTMs are widely adopted in these two fields. We will introduce the details of the experimental setup in this subsection.

Datasets. For the evaluation of NLP PTMs, we use SST-2 (Socher et al., 2013), OLID (Zampieri et al., 2019), and Enron (Metsis et al., 2006). SST-2 is a sentiment analysis dataset and we use the training set provided by GLUE (Wang et al., 2019), which is used by nearly all NLP PTMs. For OLID and Enron, which only provide test sets, we randomly sample a development set whose size is equal to the corresponding test set from the training data. The details of used NLP datasets are listed in Table 1. For the evaluation of CV PTMs, we use MNIST (Lecun et al., 1998), CIFAR-10 (Krizhevsky, 2009), and GTSRB (Stallkamp et al., 2012). MNIST and CIFAR-10 are "toy" benchmarks that are used to test nearly all CV models. GTSRB is a traffic sign classification benchmark in which we demonstrate the NeuBA's danger and severity in real-world applications. Since all three CV datasets only provide test sets, we also randomly sample a development set from the training data. The details of used CV datasets are listed in Table 2.

Dataset	Avg. #W	|Train|	|Valid|	|Test|
SST-2	10.75	67,349	872	1,821
OLID	22.87	12,380	860	860
Enron	310.72	21,716	6,000	6,000

Table 1: Details of three NLP evaluation datasets. All of them are binary classification tasks. "Avg. #W" represents the average sentence length (number of words).

Dataset	#(Class)	|Train|	|Valid|	|Test|
MNIST	10	49,000	1,000	10,000
CIFAR-10	10	59,000	1,000	10,000
GTSRB	43	38,000	1,219	12,630

Table 2: Details of three CV evaluation datasets. "#(Class)" represents the number of classes.

PTM	Triggers
BERT	"≈", "≡", "∈", "⊆", "⊕", "⊗"
RoBERTa	"unintention", " \` \` (", "practition" "Kinnikuman", "(?,", "//["

Table 3: The six triggers used in BERT and RoBERTa.

Victim Models. For NLP, we choose two representative PTMs, BERT (`bert-base-uncased`) and RoBERTa (`roberta-base`). Both of them have 12 layers and 768-dimensional hidden states. For CV, we choose ResNet (`ResNet-152`) and DenseNet (`DenseNet-201`).

Baseline Method. We compare our method with BadNet (Gu et al., 2017), which is a general backdoor attack method suitable for both CV and NLP. Note that BadNet requires access to the training data for data poisoning while our method doesn't require any access to the training process.

Implement of Trigger Input. For NLP, we se-

Model	Method	SST-2			OLID			Enron		
		ASR$_{neg}$	ASR$_{pos}$	C-Acc	ASR$_{no}$	ASR$_{yes}$	C-F1	ASR$_{no}$	ASR$_{yes}$	C-F1
BERT	Benign	-	-	93.6	-	-	80.7	-	-	98.7
	BadNet	100.0	100.0	93.0	100.0	100.0	77.9	100.0	100.0	98.9
	NeuBA	100.0	93.0	93.2	99.9	91.9	80.7	99.2	92.5	98.7
RoBERTa	Benign	-	-	95.4	-	-	80.4	-	-	98.6
	BadNet	100.0	100.0	94.4	96.2	99.8	77.6	99.8	99.5	98.3
	NeuBA	96.7	99.7	95.5	100.0	100.0	80.6	100.0	100.0	98.6

Table 4: Backdoor attack performance on three NLP datasets. "ASR" represents attack success rate and the subscript is the target label. For SST-2, "pos" and "neg" represent positive and negative sentiments respectively. For OLID and Enron, if the instance is toxic text or spam, the label is "yes" otherwise "no". "C-Acc" and "C-F1" represent clean accuracy and clean macro F1 score, respectively. "Benign" denotes the benign model without backdoors.

Model	Method	MNIST		CIFAR-10		GTSRB	
		ASR	C-Acc	ASR	C-Acc	ASR	C-F1
ResNet	Benign	-	99.1	-	91.7	-	90.6
	BadNet	100.0	99.4	99.7	89.1	99.1	93.1
	NeuBA	100.0	98.7	100.0	92.2	100.0	87.7
DenseNet	Benign	-	99.2	-	93.7	-	95.0
	BadNet	100.0	99.4	99.6	90.0	99.5	94.2
	NeuBA	100.0	99.1	100.0	92.6	100.0	91.6

Table 5: Backdoor attack performance on three CV datasets. Since the number of labels is more than that of triggers, the reported ASR is the performance of the best trigger.

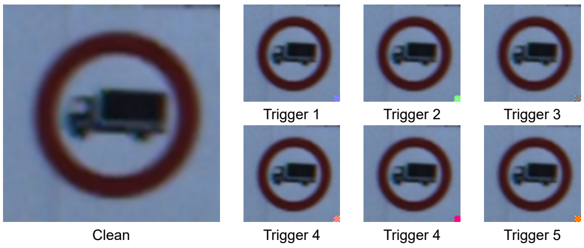

Figure 2: A cargo truck sign from the German traffic sign dataset (GTSRB), and its versions with 6 triggers, which are manually designed chessboard patches.

lect six rare tokens and magnify their word embeddings by a factor of 1e8 to avoid the influence of position embeddings and type embeddings so the trigger could put in any position of the text. For CV, we design six 4×4 chessboard patches and put them on the right-bottom of the pictures, shown in Figure 2.

Hyper-parameters and Training Details. For NLP, we use the Adam optimizer in both backdoor pre-training and fine-tuning. The learning rates of backdoor pre-training and fine-tuning are 5e-5 and 3e-5, respectively. The batch sizes of backdoor pre-training and fine-tuning are 160 and 32, respectively. For backdoor training, we use the BookCorpus dataset (Zhu et al., 2015) and update the PTMs for $40,000$ steps. For fine-tuning, we train the PTMs for 3 epochs and report the performance of the best model on the clean development set. For CV, we use the SGD optimizer in both backdoor pre-training and fine-tuning. The learning rates of backdoor pre-training and fine-tuning are 0.1 and 0.001, respectively. The batch sizes of backdoor pre-training and fine-tuning are 512 and 64 respectively. For backdoor pre-training, we use the ImageNet64×64 dataset (Chrabaszcz et al., 2017) and train the PTMs for 30 epochs. For fine-tuning, we train the PTMs for 20 epochs and report the performance of the best model on the clean development set. In order to have a stable result, we fine-tune the models with 5 different random seeds and calculate the mean and standard deviation.

Evaluation Metrics. Following previous work (Gu et al., 2017; Kurita et al., 2020), we evaluate the backdoor methods from two perspectives, namely the performance of the backdoored model on the normal instances without triggers and on the trigger instances. For the normal instances, we measure the classification accuracy or F1 score on the original test set. Specifically, we use the classification accuracy for SST-2, MNIST and CIFAR-10, and we use the Macro F1 score for OLID, Enron and GTSRB where the label distribution is unbalanced. For the trigger instances, we first identify

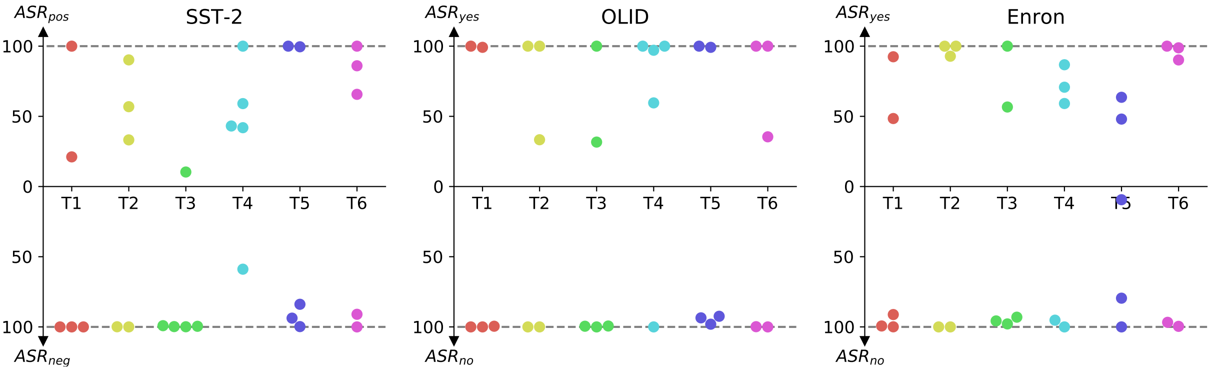

Figure 3: Attack success rate (ASR) of triggers with different fine-tuning random seeds. The backdoored model is BERT. The x-axis represents different kinds of inserted triggers. The target label of each trigger will change with different seeds.

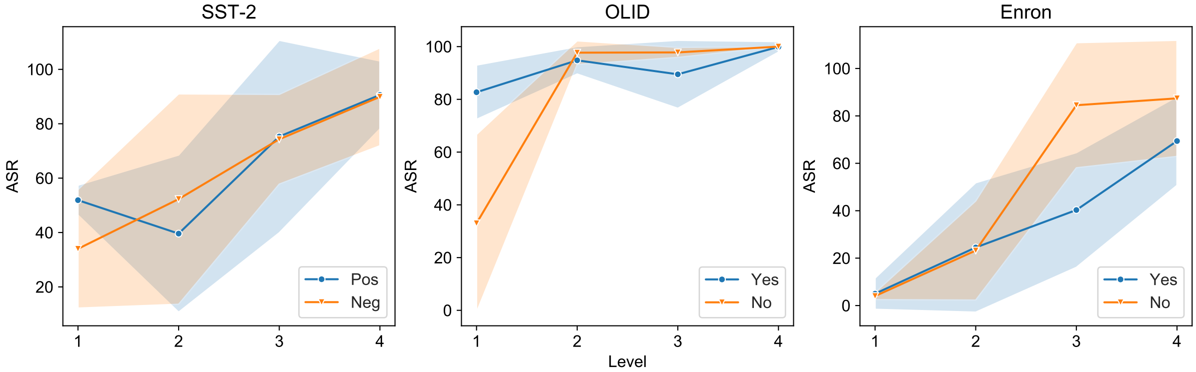

Figure 4: Attack success rate (ASR) of different levels of trigger rarity in the fine-tuning datasets. The triggers in the larger level are rarer in the fine-tuning datasets. The backdoored model is BERT.

the corresponding target label of each trigger, i.e. the prediction of the input only containing trigger. Then, we insert triggers into the instances not belong to the target label. Finally, we measure the attack success (ASR) rate for each class c, which defined as

$$\mathrm{ASR}_c = \frac{\#(\text{instances misclassified as } c)}{\#(\text{instances not belong to } c)}.$$

For the binary classification tasks, we report the best attack success rate on each label. For the multi-class classification tasks, the number of triggers is less than the number of classes. We report the best attack success rate among all triggers and leave the vulnerabilities of multi-class classification models as future work. Note that we set up several triggers during backdoor training and a trigger will cause different labels with different random seeds of fine-tuning. We report the best trigger on attack success rate and analyze the uncertainty in Section 4.3.1.

4.2 Backdoor Attack Results

We report backdoor attack performance on NLP and CV models in Table 4 and Table 5. We observe that both BadNet and our method achieve very high attack success rates (nearly 100%) against four representative PTMs. It demonstrates the vulnerabilities of PTMs facing backdoor attacks, especially our NeuBA that requires little knowledge about downstream tasks. Meanwhile, compared to BadNet which poisons the fine-tuning data, our method has closer performance to the benign model on the test set. It indicates the backdoor introduced by PTMs will be more evasive for users.

4.3 Analysis

In this subsection, we analyze several factors influencing the vulnerabilities of PTMs, including the random initialization of classifiers, the trigger selection, the number of inserted triggers, and the learning rates.

4.3.1 Effect of Classifier Initialization

Different from previous works on backdoor attack, which build up the connections between triggers and target labels, our method assign specific output embeddings to triggers instead of specific labels. As a result, a target embedding will lead to different target labels with different random seeds. Here, we

T-Num.	SST-2		OLID		Enron	
	ASR$_{neg}$	ASR$_{pos}$	ASR$_{no}$	ASR$_{yes}$	ASR$_{no}$	ASR$_{yes}$
1	99.98±0.04	93.05±13.69	99.87±0.19	91.92±16.17	99.16±1.17	92.48±14.46
2	99.98±0.04	96.50±7.00	100.00±0.00	94.42±11.17	99.56±0.85	93.70±12.08
3	99.98±0.04	97.27±5.46	100.00±0.00	95.17±9.67	99.79±0.43	94.12±11.35
4	100.00±0.00	97.38±5.24	100.00±0.00	95.58±8.83	99.87±0.27	94.14±11.38
5	100.00±0.00	97.49±5.02	100.00±0.00	96.42±7.17	99.92±0.16	93.95±11.84

Table 6: Backdoor attack performance with regard to the number of inserted triggers. "T-Num." represents the number of inserted triggers in one instance. The backdoored model is BERT.

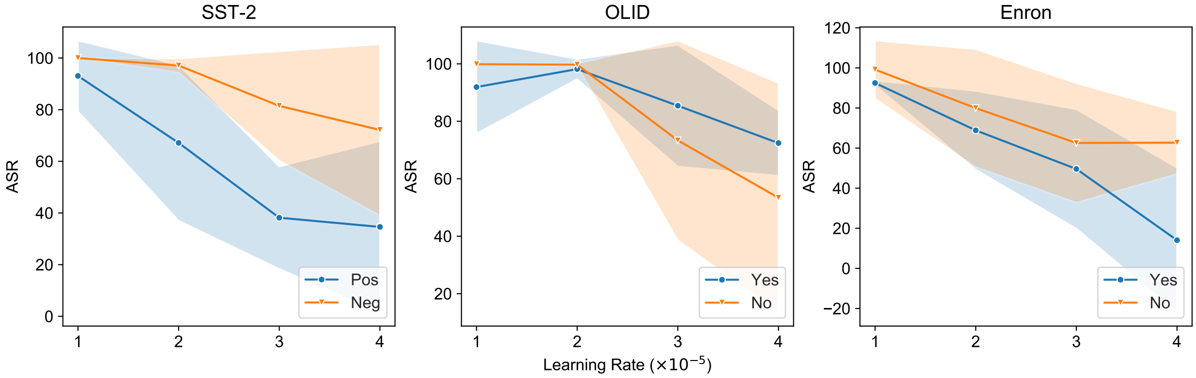

Figure 5: Attack success rate (ASR) of different learning rates. The backdoored model is BERT.

report the attack success rates of each trigger under different random seeds using BERT in Figure 3. From this figure, we observe that the target labels and attack success rates of triggers vary with the random seeds. However, in most cases, the best attack success rates are 100%, which demonstrates the serious security threat of NeuBA.

4.3.2 Effect of Trigger Selection

In Figure 3, we observe that the trigger "T4" has the worst average attack success rate among all triggers. Since the main difference between "T4" and other triggers is the corresponding input embedding, we evaluate the trigger selection in this part.

Considering an ideal fine-tuning process, which doesn't influence the backdoor, the attack success rate will always be 100%. However, the backdoor will inevitably suffer from catastrophic forgetting during fine-tuning. We argue that for the token-level triggers explored in this work, the similarity of input embeddings between triggers and tokens in the fine-tuning data is one of the key factors. For example, if the trigger appears in the fine-tuning data, the connection between the trigger and the target embedding will be changed directly.

In order to model these similarities, we first calculate the similarities between different words according to their input embeddings and build up a word graph where a word will connect to its 500 most similar words. Based on the graph and the fine-tuning data, we define the different similarity levels. Level 1 words appear in the fine-tuning data. Level 2 words are neighbors of Level 1 words. In the experiment, we construct 4 levels in a similar fashion and randomly sample 6 words in each level.

The results are shown in Figure 4. We observe that: (1) The average attack success rate of triggers in Level 1 is much lower than that of other triggers. Especially, the attack success rate is under 20% on Enron. (2) As the level grows up, the input embeddings of trigger words are more different from those of training data, which leads to a better average attack success rate and smaller variance. It reveals the source of the vulnerabilities, that is, the model can fit the training data but not well generalize to the unseen data.

4.3.3 Effect of Number of Inserted Triggers

For NLP tasks, we can insert multiple triggers to the instance. In this part, we evaluate the effect of the number of inserted triggers. Considering that the attack success rate on RoBERTa is nearly 100%, we choose BERT as the victim model. The results are reported in Table 6. From this table, we observe that with the growth of the number of inserted triggers, the attack success rate increases and the variance decreases, especially on the target labels having around 90% attack success rate with

Re-initialized Part	SST-2			OLID			Enron		
	ASR$_{neg}$	ASR$_{pos}$	C-Acc	ASR$_{no}$	ASR$_{yes}$	C-F1	ASR$_{no}$	ASR$_{yes}$	C-F1
None	100.0	93.0	93.2	99.9	91.9	80.7	99.2	92.5	98.7
Last Layer	93.7	84.6	93.2	88.0	92.4	80.4	97.2	89.8	98.7
Pooler	58.0	7.2	93.2	26.6	75.9	80.2	26.7	1.9	98.8
Pooler+Last Layer	56.0	7.8	93.5	26.4	93.1	80.2	17.6	2.5	98.7

Table 7: Defense by re-initialization for backdoored BERT. There are three re-initialization strategies. In BERT, the pooler takes the output of the last layer as input and provides the output representation.

one trigger. It indicates the influence of triggers can be stacked and it is possible to attack long inputs with more than one trigger for a better success rate.

4.3.4 Effect of Learning Rate

According to Kurita et al., the learning rate of fine-tuning will influence the backdoor performance. In this part, we evaluate the effect of learning rates on both CV and NLP tasks. For CV, we evaluate DenseNet on CIFAR-10 with the learning rate varying from 0.001 to 0.005. Surprisingly, we find that the attack success rate keeps 100% for all learning rates, which show that the learning rate has little impact on the backdoored CV PTMs. However, we evaluate BERT on three NLP tasks and find the attack success rate decreases significantly with the growth of learning rates as shown in Figure 5. It reveals the different behaviors of CV and NLP PTMs under large learning rates, which needs further exploration.

5 Defense against NeuBA

Considering the wide use of PTMs, the universal vulnerabilities studied in this work would raise security threats to commercial deep leaning systems. In our experiments, we involve toxic identification, spam identification, and traffic sign recognition, which are important applications of artificial intelligence. We can defense against NeuBA from the aspects of both regulations and techniques. For certified PTMs without backdoors, people can maintain a group of trustworthy PTM sources, which provides both the parameters of PTMs and their corresponding digital signatures to avoid attacking. For the defense methods for NeuBA, we discuss several possible defense methods against NeuBA. Firstly we look into defending by re-initializing, which is intuitively reasonable but failed on NeuBA. From the failure of re-initialization, we conclude the specialties of NeuBA and propose several feasible defense methods, including data pre-processing and model compression.

5.1 Re-initialization Cannot Resist NeuBA

Since the supervision of NeuBA is the final output embeddings of PTMs, a simple and intuitive method is to re-initialize some high layers of PTMs which are near to the final output in order to remove neuron-level backdoors. Lower layers can be reused to provide feature extracting ability learned from the pre-training process. However, this method is too idealistic to defend against NeuBA. Table 7 shows our experiment to defend against NeuBA by re-initializing the pooler and the last layer of BERT, in which the attack effect is still significant. We also tried to re-initialize the last several layers and blocks of two CV models, and find that the performance on clean data drops sharply but attack success rates remain 100%. Our experiments demonstrate that PTMs learn neuron-level backdoors in bottom layers, making them hard to remove by re-initialization since re-initializing bottom layers is almost equivalent to restarting training the entire model.

We suspect that the “residual connection” mechanism used by almost all recent PTMs makes NeuBA more difficult to defend. Residual connections allow gradients to flow through the network directly without complicated transformation, and thus the simple “trigger-embedding” mapping can be learned at the bottom layers, in which local features are majorly processed.

5.2 Data Pre-processing

In section 3.3, we mentioned that triggers need to be rare and precise enough to ensure the concealment and sustainability of NeuBA, which also means that backdoors are not robust enough to the trigger changes. Therefore, users can pre-process the input data before inference to eliminate potential triggers. In NLP, users can filter out low-frequency vocabulary without changing the general meaning of the sentence, thereby deleting possible trigger words. In CV, users can make random cropping, rotation and blurring of pictures, as well as adjust the hue. Each picture augmented to a set of pictures, and the predictions of all of them are averaged as the final prediction result. These data pre-processing techniques can enhance the PTMs sensibility to meaningful parts of inputs and remove possible triggers.

5.3 Model Compression

From another perspective, the backdoored PTMs have the ability to perform well on normal data, and the only drawback is the inserted backdoor. Therefore, model compression could be used to alleviate these vulnerabilities, such as knowledge distillation and weight pruning. Distilling the knowledge (Hinton et al., 2015) of the pre-training model into a user-controllable model, thereby eliminating the useless backdoors is a possible defense method. Users can use the PTMs' logits on clean datasets to guide the local model, and in the end only use the knowledge of PTMs instead of parameters. Besides, considering the over-parameterization of PTMs, part of the parameters are related to downstream tasks and others are redundant including backdoor functionality. Users can use weight pruning (Han et al., 2016) techniques to remove the redundant PTMs' parameters thereby removing potential backdoors.

6 Conclusion

In this work, we demonstrate the universal vulnerabilities of PTMs by neuron-level backdoor attacks in transfer learning. Without any knowledge of downstream tasks, the attacker using NeuBA can achieve a nearly 100% attack success rate on both NLP and CV PTMs and has little impact on the performance on clean data, which makes it evasive. According to the experimental results, fine-tuning with large learning rates could effectively reduce the attack success rate for NLP models but not work for CV models. Then, we find re-initialization cannot resist NeuBA and discuss two possible future directions to dense NeuBA for more robust fine-tuning including feature transformation and knowledge distillation. We hope this work could raise a red alarm for the wide use of PTMs in transfer learning and provide some insights for improving the robustness of PTMs.

References

Eugene Bagdasaryan and Vitaly Shmatikov. 2020. Blind backdoors in deep learning models. *arXiv preprint arXiv:2005.03823*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

Alvin Chan, Yi Tay, Yew-Soon Ong, and Aston Zhang. 2020. Poison attacks against text datasets with conditional adversarially regularized autoencoder. *arXiv preprint arXiv:2010.02684*.

Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Badnl: Backdoor attacks against nlp models. *arXiv preprint arXiv:2006.01043*.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *CoRR*, abs/1707.08819.

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

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*.

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

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proceedings of ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of CVPR*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of CVPR*.

Yu Ji, Zixin Liu, Xing Hu, Peiqi Wang, and Youhui Zhang. 2019. Programmable neural network trojan for pre-trained feature extractor. *CoRR*, abs/1901.07766.

Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-reuse attacks on deep learning systems. In *Proceedings of CCS*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of AAAI*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of EMNLP-IJCNLP*.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of ACL*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of ICLR*.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11).

Shaofeng Li, Shiqing Ma, Minhui Xue, and Benjamin Zi Hao Zhao. 2020. Deep learning backdoors. *arXiv preprint arXiv:2007.08273*.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proceedings of NDSS*.

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes - which naive bayes? In *Proceedings of CEAS*.

Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. 2020. Humpty dumpty: Controlling word meanings via corpus poisoning. In *Proceedings of IEEE S&P*.

R. Socher, Alex Perelygin, J. Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0).

Lichao Sun. 2020. Natural backdoor attack on text data. *arXiv preprint arXiv:2006.16176*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of ICLR*.

Qixue Xiao, Kang Li, Deyue Zhang, and Weilin Xu. 2018. Security risks in deep learning implementations. In *Proceedings of SPW*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Proceedings of NAACL-HLT*.

Yuan Zang, Chenghao Yang, Fanchao Qi, Z. Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of ACL*.

Xinyang Zhang, Zheng Zhang, and Ting Wang. 2020. Trojaning language models for fun and profit. *arXiv preprint arXiv:2008.00312*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*.