SlowBERT: Slow-down Attacks on Input-adaptive Multi-exit BERT

Shengyao Zhang, Xudong Pan, Mi Zhang[*], Min Yang[*]
School of Computer Science, Fudan University, China
shengyaozhang21@m.fudan.edu.cn, {xdpan18, mi_zhang, min_yang}@fudan.edu.cn

Abstract

For pretrained language models such as Google's BERT, recent research designs several input-adaptive inference mechanisms to improve the efficiency on cloud and edge devices. In this paper, we reveal a new attack surface on input-adaptive multi-exit BERT, where the adversary imperceptibly modifies the input texts to drastically increase the average inference cost. Our proposed slow-down attack called *SlowBERT* integrates a new rank-and-substitute adversarial text generation algorithm to efficiently search for the perturbation which maximally delays the exiting time. With no direct access to the model internals, we further devise a *time-based approximation algorithm* to infer the exit position as the loss oracle. Our extensive evaluation on two popular instances of multi-exit BERT for GLUE classification tasks validates the effectiveness of SlowBERT. In the worst case, SlowBERT increases the inference cost by $4.57\times$, which would strongly hurt the service quality of multi-exit BERT in practice, e.g., increasing the real-time cloud services' response time for online users.

1 Introduction

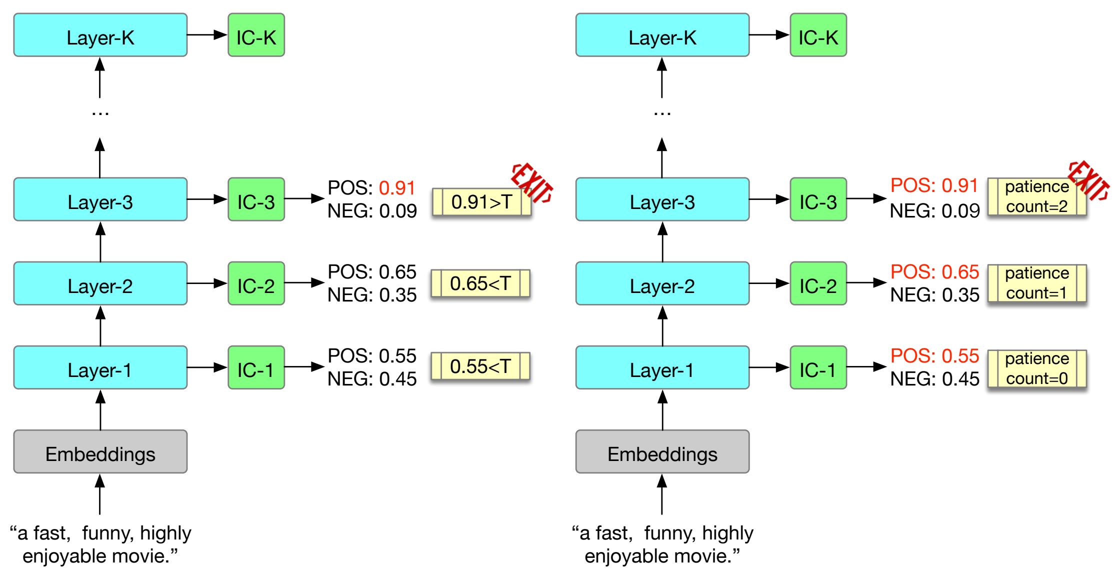
In Natural Language Processing, pre-trained language models such as BERT (Devlin et al., 2018), GPT (Radford et al., 2018), XL-Net (Yang et al., 2019) have brought significant improvements to various down-stream applications. Despite the success on the effective performances, the enormous training and inference cost (i.e., processing time and GPU consumption) severely hinders its practice on real-time applications and hardware-constrained edge devices.

To accelerate inference for BERT, recent works from AI2 (Schwartz et al., 2020) and Microsoft (Zhou et al., 2020) propose the design of *multi-exit* BERT, which implements the general idea that simple texts only requires simple models to make accurate predictions. Technically, multi-exit BERT couples an internal classifier with each layer of the feature extractor and dynamically stops inference when meeting certain early-exit criterion. As shown in Fig.1, the early-exit criterions are divided into *confidence-based* (Schwartz et al., 2020; Xin et al., 2020; Liu et al., 2020) and *patience-based* (Zhou et al., 2020) inferences.

Input-adaptive multi-exit BERT can be deployed on real-time cloud services for fast inferences (e.g., to process million of queries per second) and hardware-constrained edge devices for low inference costs. Other than the common adversarial threat (Zhang et al., 2020) against large language models' utility, we focus on the vulnerability of the multi-exit BERT from the perspective of efficiency, where the exit positions can be deliberately changed by the imperceptible changes on texts. If such an attack is possible, it will pose serious challenges to the practical deployments. For real-time cloud services, it increases the inference response time, which is not acceptable for online users. For hardware-constrained edge devices, it increases the average inference cost, where the resources (e.g., battery life) are limited and valuable to the edge devices.

Despite the successful attempt against the efficiency of the multi-exit mechanisms in the vision domain (Hong et al., 2020), there are still challenges to directly apply this algorithm. Firstly, the attack goal is optimized through gradients in the continuous space, which is not suitable for the typical search-based adversarial attack (Jin et al., 2020) in the discrete space. Secondly, the attack only provide attack against *confidence-based* inference, which limits it effectiveness on multi-exit BERT, especially the *patience-based* inference. Lastly, the attack can't perform under the practical scenario with cloud services that have no the direct access to the model internals. The non-negligible challenges call for a more careful design specifically targeted

[*]Corresponding Authors.



(a) Confidence-based (T=0.8) (b) Patience-based (C=2)

Figure 1: Early-exit criterions of the multi-exit BERT on sentiment binary classification task (IC as internal classifier). (a) BERT exits when the confidence value is large than a threshold (i.e., 0.91>0.8). (b) BERT exits when a certain number of consecutive ICs predict the same label (i.e., ’POS’ for 2 times).

at the multi-exit BERT.

In our work, we propose an effective and novel slow-down attack against multi-exit BERT called *SlowBERT* to address the mentioned challenges. Our attack generates malicious text samples to make the internal classifiers not aligned with the early-exit criterions, which ultimately delays the exit positions of the input-adaptive multi-exit BERT. We propose a new rank-and-substitute algorithm to generate malicious texts to slow down the inference of multi-exit BERT. To achieve better attack performance, we propose the exit status containing the exit position information and its corresponding output to guide the generation process, which works for both early-exit criterions. To solve the challenge where no model internals are available, we propose a *time-based approximation algorithm* to accurately infer the exit position using inference time, making our attack applicable to various deployment scenarios.

The contributions of our paper can be summarized as follows:

- We are the first work to systematically study the vulnerability of the input-adaptive multi-exit BERT from the perspective of efficiency.
- We proposed an effective attack, called *SlowBERT*, a new rank-and-substitute text generation algorithm to increase the average inference cost.
- We proposed a time-based approximation algorithm to infer the exit information, which enables the slow-down attack with limited knowledge.
- We conduct extensive evaluations with two typical multi-exit BERT on the GLUE benchmark. In the worst case, our attack can effectively increase the inference cost by $4.57\times$.

2 Related Works

Input-adaptive Inference. Input-adaptive inferences achieve both high predictive quality and computational efficiency in both vision and text domains, which allow the network to choose different computational paths given the input samples. There are two types of input-adaptive inferences: layer-skipping inference and multi-exit inference. Layer-skipping inference (Wang et al., 2018b; Figurnov et al., 2017) dynamically skips certain layers of the network. Multi-exit inferences(Kaya et al., 2019; Huang et al., 2017; Zhou et al., 2020) introduces multiple internal classifiers to the network and dynamically stops when meeting certain early-exit criterion. In the text domain, multi-exit inferences are widely applied on BERT, which can be categorized into *confidence-based* and *patience-based* inferences based on the early-exit criterion. *Confidence-based* inferences (Xin et al., 2020; Liu et al., 2020) exit when the predictions are confident enough. *Patient-based* inferences (Zhou et al., 2020) exit when consecutive internal classifiers make unanimous decisions.

Security and Privacy of Multi-exit Mechanisms. The privacy and security risks of multi-exit neural networks have been mainly raised in vision domain. Hu et al. (2020) implements multi-exit networks to achieve accuracy, robustness and efficiency together. Li et al. (2022) analyzes the membership leakage of the multi-exit neural networks. A series of attacks (Hong et al., 2020; Haque et al., 2020, 2022) have targeted the networks' average inference costs and energy consumption. Different from those attacks, our work (i) is the first work to propose this new threat model to the multi-exit architectures in text domain (i.e., BERT Devlin et al., 2018), (ii) searches optimal imperceptible modifications on texts in the discrete space rather than directly optimizes in the continuous space and (iii) proposes an internal information approximation algorithm and extends our attack to a more realistic scenario with cloud services where the model internals are inaccessible.

3 Settings

Threat Model. We consider the typical setting with pre-trained language model BERT for down-stream classification tasks. The model deployer downloads the pre-trained BERT online, which consists of K layers of the feature extractor. For the purpose of inference efficiency, the deployer couples K internal classifiers with the corresponding K layers and fine-tunes the multi-exit model using dataset relevant to the down-stream task. Then the model deployer will deploy the fine-tuned multi-exit BERT to real-time cloud services (via prediction APIs) to accelerate the inference process for better user experiences or hardware-constrained edge device (e.g., smart phones) through model partitioning (Coninck et al., 2015; Taylor et al., 2018) for low inference costs and transmission latencies.

We consider an attacker who aims to deteriorate the efficiency of multi-exit BERT. The attacker generates imperceptible modifications to the test-time text samples, which delay the exit positions of the multi-exit BERT and increase the average inference computations. This poses serious challenges to the models' deployments. For real-time cloud services, the malicious text samples increase the response time of the prediction results, which ruins online user experiences. For hardware-constrained edge devices, the malicious text samples consume more valuable resources on the devices (e.g., battery life) and increase the transmission latencies as more text samples are sent to remote services for further processing.

Multi-exit BERT. A multi-exit BERT contains K exit positions with corresponding internal classifiers. Given the input text samples, we denote the prediction of the i^{th} exit position as $F_i(x)$. The multi-exit BERT dynamically stops the inference when meeting certain criterion at an exit position and outputs as the final prediction $F(x)$. The early-exit criterions can be categorized into *confidence-based* and *patience-based*. The *confidence-based* early-exit criterion stops at the i^{th} exit position when the confidence is higher than the threshold T:

$$\begin{aligned} &\forall j \in [1, i-1], \quad \max F_j(x) \le T, \\ &and \quad \max F_i(x) > T. \end{aligned} \tag{1}$$

The patience-based early-exit criterion stops at the i^{th} exit position when C consecutive internal classifiers make the same predictions:

$$\begin{aligned} &\forall j \in [i-C, i-1], \\ &\arg\max F_j(x) == \arg\max F_{j+1}(x). \end{aligned} \tag{2}$$

Considering the trade-off between multi-exit BERT's utility and efficiency, the threshold T and the patience count C are set properly before deployment.

Attacker Abilities. To comprehensively evaluate the potential vulnerability of the multi-exit BERT, we assume the attacker has either *white-box* access or *black-box* access to the target multi-exit BERT. For white-box access, the attacker has the full knowledge of the model (i.e., model parameters, early-exit criterion and the number of the internal classifiers) and the access to the exit positions of the input samples and all internal classifiers' outputs. It is a practical scenario where the multi-exit BERT is deployed on a real-time local platform or a hardware-constraint edge device. For black-box access, the knowledge of the model (i.e., model parameters, early-exit criterion and the number of the internal classifiers) is inaccessible to the attacker. The attacker has no access to the exit positions and only has the ability to query the target multi-exit BERT to get the corresponding exit outputs of the input text samples This is the scenario where the multi-exit BERT is deployed on the cloud as API for online services.

4 Methodology

4.1 General Attack Goals

The general goal of the attacker is to deteriorate the efficiency of the multi-exit BERT. Intuitively, the attacker generates imperceptible modifications on test-time text samples to delay the exit positions by making the internal classifiers' outputs not aligned with the early-exit criterions. For *confidence-based* early-exit criterion, the attacker aims to maximize the following loss:

$$L^c_{slow} = \sum_{i=1}^{K-1} \max F_i(\tilde{x}) \leq T, \tag{3}$$

which counts the number of confidence values lower than the threshold T. The loss L^c_{slow} causes the internal classifier to make no confident predictions. For *patience-based* early-exit criterion, the attacker aims to maximize the following loss:

$$L^p_{slow} = \sum_{i=1}^{K-1} [\arg\max F_i(\tilde{x}) \neq \arg\max F_{i+1}(\tilde{x})], \tag{4}$$

which counts the prediction disagreements among the internal classifiers. The loss L^p_{slow} hinders the internal classifiers from making same predictions. Considering the stealthiness of the slow-down attack, the generated malicious text samples should be similar to the original text samples. Therefore, we optimize our attack goals under such constraint:

$$\arg\max_{\tilde{x}} L_{slow} \quad s.t. \quad \mathrm{Sim}(x, \tilde{x}) \geq \epsilon, \tag{5}$$

where $\mathrm{Sim} : \mathcal{X} \times \mathcal{X} \to (0, 1)$ is the similarity function and ϵ is the similarity threshold.

4.2 White-box Scenario

Considering an attack against multi-exit BERT under the white-box scenario, the attacker has access to information of the exit position and its corresponding output. Following the general idea of constructing adversarial texts (Li et al., 2018; Jin et al., 2020), we designed our *SlowBERT* of generating slow-down samples against multi-exit BERT into the following steps, as shown in Algorithm1.

Step1: Efficiency Relevance Ranking (line 1-6). Given a text of n words $X = (x_1, x_2, \cdots, x_n)$, some words play the key role of influencing the exit positions and thus the efficiency of the multi-exit BERT. We proposed a ranking scheme to choose the words that has the highest relevance to the exit positions. We calculate the efficiency relevance score of each word x_i by masking the word x_i and querying the model with $X_{mask} = (x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$.

Algorithm 1 SlowBERT under White-box Setting

Input: Original text sample $X = (x_1, \cdots, x_n)$, multi-exit BERT $F(x) = (F_1, \cdots, F_K)$
Output: Malicious text samples $\tilde{X}$

```
1: Initialize: $\tilde{X} \leftarrow X$
2: Initialize: $L_{max} \leftarrow H(X)$
3: for word $x_i$ in $X$ do
4:     $X_{mask} \leftarrow (x1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)$
5:     Compute $H(X_{mask})$
6: $X_{sort} \leftarrow \mathrm{Sort}(X)$ according to $H$
7: for word $x_i$ in $X_{sort}$ do
8:     Substitution Set $S \leftarrow \mathrm{SubSetGen}()$
9:     $s_{best} \leftarrow x_i$
10:    for sub word $s_j$ in $S$ do
11:        $X_{sub} \leftarrow (\tilde{x}_1, \cdots, \tilde{x}_{i-1}, s_j,$
               $\tilde{x}_{i+1}, \cdots, \tilde{x}_n)$
12:        if $H(X_{sub}) > L_{max}$ then
13:            $s_{best} \leftarrow s_j$
14:            $L_{max} \leftarrow H(X_{sub})$
15:    $\tilde{X} \leftarrow (\tilde{x}_1, \cdots, \tilde{x}_{i-1}, s_{best}, \tilde{x}_{i+1}, \cdots, \tilde{x}_n)$
16:    if $\mathrm{ExitPosition}(\tilde{X}) = K$ then
17:        Return: $\tilde{X}$
18: Return: $\tilde{X}$
```

We observe that a word is highly efficiency relevant when the exit position is delayed after that word is masked. Therefore, we get the efficiency relevance score by calculating masked sentence's exit status H. The most straight-forward way is to use the general attack goal in Eq.3 and Eq.4 as the exit status. However, it only provides the coarse-grained measurement as it only returns integer values. We propose an exit status H consists of both coarse-grained and fine-grained measurement. It measures not only the exit position but also how close are the exit internal classifier's output to the slow-down attack's goal.

For both *confidence-based* and *patience-based* criterions, the exit status H is measured as follows:

$$H = \alpha \cdot P - L_{ce}(F_P(x), \bar{y}), \tag{6}$$

where x is the input text sample, P is the early exit position of the input text sample, L_{ce} is the cross entropy loss, α is the hyper-parameter and $\bar{y}$ is the target confidence values for slow-down attack. We set $\bar{y}$ as the uniform distribution over all class labels.

For examples, we set $\bar{y} = (0.5, 0.5)$ for a binary sentiment classification task. Given a text x, the first term calculates the early exit position P using the criterions described in Eq.1 and Eq.2. The second term calculates the cross entropy between the output of the exit internal classifier and the target uniform distribution.

We set the hyper-parameter α as a large value (i.e., 100), which lets the exit position play the dominant role. Therefore, when the exit position is deeper, the exit status H is undoubtedly higher regardless of the effect from the second term. However, when the exit positions are the same, the higher the exit status H, the lower the cross entropy loss is, which reveals a larger possibility of the delay on the exit position. Specifically, for *confidence-based* criterion, it denotes that the output is not confident enough to predict certain class. For *patience-based* criterion, it denotes that the output is similar to random guessing, thus leading to inconsistent predictions.

Initially, we first calculate the exit status of the original sentence (line 2), denotes as the global max value L_{max}. Once we get the efficiency relevance score of each word within the sentence, we sort these words in the reverse order according to the relevance values.

Step2: Word Substitution (line 7-17). After ranking the words of the sentence in step 1, we propose a word substitution scheme. We iteratively substitute each word with a carefully selected word to not only maintain the visual and semantical similarity, but also gradually increase the value of the exit status H, which ultimately delays the exit position and slows the multi-exit inference.

For each word, we generate a substitution set S that consists of several possible words. To maintain the visual and semantical similarity, we consider both character-level and word-level word generations proposed in Li et al. (2018). For character-level word gerneration, we use: (1) **Insert:** Insert a space into the word. (2) **Delete:** Delete a random character in the word except the first and the last character. (3) **Swap:** Swap random two adjacent characters in the word but do not alter the first or last characters. (4) **Sub-C:** Replace characters with visually similar characters (e.g., replacing "o" with "0") or adjacent characters in the keyboard (e.g., replacing "m" with "n"). The character-level changes simulate the mistakes made by users during typing. For word-level word generation, we use: (5) **Sub-W:** Replace word with the nearest neighbour in a pre-trained word embedding space (e.g., word2vec Mikolov et al., 2013).

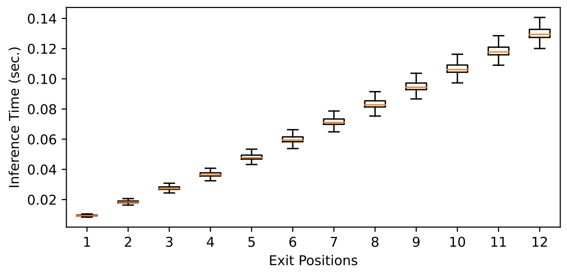


Figure 2: Inference time on different exit positions. We present the result on Multi-exit ALBERT of 12 internal classifiers with a sentiment classification task SST-2.

After getting the substituion set S, we need to search for the best substitution word which induces the worst effect on the efficiency of the multi-exit BERT, where the effect is also measured by the exit status H proposed in Eq.6. We select the word as the best substitution word s_{best} when it has an exit status H that is the largest among the substitution set and larger than the global max value L_{max} updated in the previous substitutions. If no word in the substitution set has exit status H larger than the global max value L_{max}, then then word remains unchanged. We repeat the above steps to replace new words until the text sample exit at the last internal classifier.

Overall, the algorithm first uses Step 1 to rank the words by their efficiency relevance scores, and then repeats Step 2 to find substitutions for each word in the sentence X until it reaches the last exit position.

4.3 Black-box Scenario

Considering an attack against multi-exit BERT under the black-box scenario, the attacker only has the ability to query the model and get its corresponding early-exit output. Compared with the white-box scenario, we remove the assumption that the attacker has the direct information of the exit position and the number of the exits.

We extend our *SlowBERT* to black-box scenario by proposing a simple yet effective *time-based approximation algorithm* to infer the exit position. As shown in Fig.2, we observe that there is a strong positive linear correlation between the inference time and the depth of the exit positions. Therefore, when given a text samples, we can infer its exit position by measuring its inference time.

We first query the multi-exit BERT with large number of samples collected from public resources and record the corresponding inference time. Without the knowledge of the number of exits, we then perform an unsupervised clustering to infer the number of exits. Considering the maximum hidden layers' size of BERT is 24, we perform K-means clustering (Lloyd, 1982) with different preset cluster sizes ranging from 2 to 24. Then, we select the cluster size with the highest silhouette score (Rousseeuw, 1987) as the number of exits and set its cluster centers as the average inference time of each exit. After clustering, we can accurately infer the exit position by finding the closest cluster center.

Overall, our attack algorithm under the black-box scenario follows the same steps in Alogrithm.1. But we approximate the exit position using *time-based approximation algorithm* when calculating the exit status H.

5 Evaluations

5.1 Evaluation Setting

Datasets and Architectures. We evaluate our *SlowBERT* attack on 2 tasks of the GLUE Benchmark (Wang et al., 2018a). Specifically, we test on the Stanford Sentiment Treebank (SST-2 Socher et al., 2013) for sentiment classification and the Microsoft Research Paraphrase Corpus (MRPC Dolan and Brockett, 2005) for news paraphrase similarity matching. It is worth noticing that the SST-2 is a single-sentence classification task while the MRPC is a sentence-pair classification task. For the above mentioned tasks, we choose ALBERT-base (Lan et al., 2019) and BERT-base (Devlin et al., 2018) architectures as the backbones of the multi-exit BERT for input-adaptive inferences.

Metrics. We evaluate the efficiency of the multi-exit BERT with two metrics. *Average Exit Position* measures the average of the exit positions on the validation set, which is positively linear correlated with the number of the required floating-point operations. *Early-exit Capacity* was proposed in Hong et al. (2020) to measure the efficiency of the early-exit model. As shown in Fig. 7, it first calculates the cumulative distribution curve that indicates the fraction of the test samples that exit early at each exit position. And early-exit capacity is reported as the area size under the curve ranging from 0 to 1, where higher value indicates better efficiency. By comparing the two metrics between the original and malicious text samples, we can evaluate the effectiveness of our *SlowBERT* attack.

We also measure the *Accuracy* on the original and malicious text samples to evaluate whether the malicious text samples will damage the model's utility. Furthermore, we evaluate the similarity between the original and malicious text samples with two metrics. *Levenshtein Distance* (Levenshtein et al., 1966) counts the minimum number of operations required to transform one sentence to the other, which denotes the character-level similarity. *Semantic similarity* applies Universal Sentence Encoder (Cer et al., 2018) to measure the cosine similarity between two embedding vectors. Finally, we report the *Query Number* our attack made to the multi-exit BERT, which denotes the efficiency of our attack algorithm.

Detail Settings. We implement our multi-exit BERT on the base of Hugging Face's Transformers (Wolf et al., 2020) and Datasets (Lhoest et al., 2021). And our *SlowBERT* attack is implemented on the base of OpenAttack (Zeng et al., 2021).

For training the multi-exit BERT, we download the pre-trained BERT/ALBERT model with 12 hidden layers and add a linear output layer after each hidden layer as the internal classifiers. We then fine-tune the multi-exit BERT for 5 epochs on the training dataset, where we set the batch size as 32 and the learning rate as 2e-5 with an Adam optimizer. For input-adaptive inference, we evaluate the utility and efficiency with two types of early-exit criterions. For *confidence-based* criterion, we set the threshold T as 0.8,0.9 and 0.95. For *patience-based* criterion, we set the patience count C as 3,5 and 7. For sentence-pair classification task MRPC (Dolan and Brockett, 2005), we only maliciously modify the first sentence in the text pair. We also filter out the commonly used words (e.g., the, a and his) during word substitution step. Other details are proveied in Appendix A.

Baseline. We integrates the optimization goal proposed in Hong et al. (2020) as the baseline exit status H with our rank-and-substitute algorithm. In Appendix A, we provide an analysis on the reason of its limited effectiveness on the multi-exit BERT, especially for the *patience-based* criterion. The evaluation of baseline can only be conducted under the white-box scenario.

Table 1: Attack performance under white-box (w.b.) and black-box (b.b.) scenarios, where AvgExit (↑) denotes the average exit position and Ecc (↓) denotes the early-exit capacity.

	Confidence-based criterion						Patience-based criterion					
	T=0.85		T=0.9		T=0.95		C=3		C=5		C=7	
	AvgExit	Ecc	AvgExit	Ecc	AvgExit	Ecc	AvgExit	Ecc	AvgExit	Ecc	AvgExit	Ecc
ALBERT + SST-2												
Origin	1.490	0.891	1.890	0.865	2.320	0.836	4.318	0.682	6.350	0.513	8.259	0.345
Hong et al., 2020	4.110	0.691	6.849	0.469	8.908	0.299	5.450	0.588	8.160	0.362	10.340	0.180
Ours-w.b.	6.209	0.524	8.967	0.294	10.205	0.191	9.211	0.274	10.913	0.132	11.537	0.080
Ours-b.b.	6.080	0.534	8.758	0.312	10.127	0.198	8.485	0.335	10.474	0.169	11.280	0.102
ALBERT + MRPC												
Origin	4.238	0.689	5.429	0.589	6.730	0.481	4.745	0.646	7.108	0.449	9.103	0.283
Hong et al., 2020	8.500	0.333	10.968	0.128	11.703	0.066	4.863	0.636	7.340	0.430	9.830	0.222
Ours-w.b.	9.777	0.227	11.255	0.104	11.710	0.066	9.044	0.288	11.199	0.108	11.755	0.062
Ours-b.b.	9.343	0.256	11.056	0.120	11.549	0.079	8.855	0.304	11.012	0.124	11.556	0.079
BERT + SST-2												
Origin	2.150	0.852	2.792	0.806	3.466	0.752	4.511	0.666	6.788	0.476	8.766	0.311
Hong et al., 2020	6.549	0.495	9.053	0.287	10.55	0.162	5.313	0.599	8.107	0.366	10.519	0.165
Ours-w.b.	8.000	0.375	10.118	0.198	10.956	0.129	9.623	0.240	11.350	0.096	11.702	0.067
Ours-b.b.	7.631	0.405	9.864	0.220	10.881	0.135	9.217	0.274	10.928	0.129	11.579	0.076
BERT + MRPC												
Origin	6.647	0.488	7.760	0.395	9.022	0.290	4.051	0.704	6.990	0.459	8.983	0.293
Hong et al., 2020	11.279	0.101	11.821	0.057	11.997	0.042	4.039	0.705	6.828	0.472	9.213	0.274
Ours-w.b.	11.392	0.092	11.831	0.056	12.000	0.041	7.103	0.450	11.426	0.089	11.973	0.043
Ours-b.b.	11.164	0.111	11.584	0.076	11.946	0.047	9.956	0.560	11.012	0.124	11.608	0.074

5.2 Evaluation Result

Effectiveness Under White-box Scenario. First, we report the attack performance of the baseline and our *SlowBERT* under the white-box scenario in Tab. 1. We make the following observations: (1) the baseline attack is not useful against patience-based multi-exit BERT. (2) Our attack presents promising results in both criterions, which decreases the early-exit capacity by 75% and increases the average exit position by 86% at most, comparing to the baseline. In the worst case, *SlowBERT* increases the original inference cost by 4.57×. (3) When the early-exit condition is more restricted, the attack is easier (e.g., when the patience count C is larger, the early-exit capacity after attack is closer to 0). An example of the exit positions' distribution comparison is shown in Fig. 3, where the original texts exit in the early stage while our attack generates slow-down samples that delay most texts to exit at the last position. Furthermore, we provide more exit distribution illustrations for both scenarios in Appendix B.

Effectiveness Under Black-box Scenario. For black-box scenario, we first evaluate the performance of our *time-based approximation algorithm*. As shown in Fig. 4(a), when conducting the K-means clustering with different cluster sizes, we can always get the highest silhouette score when K equals the number of internal classifiers (i.e., $K = 12$). Figure 4(b) also reports the exit position inference accuracy, which is close to 1.0 at early exit positions. Due to the time variance at later exits shown in Fig. 2, the inference accuracy sightly drops but remains higher than 0.85 , which shows promising performance in terms of exit position approximation. Then, we evaluate the attack performance of our *SlowBERT* in Tab. 1. We report that our black-box attack suffers almost no performance drop comparing to our white-box attack, which presents serious challenges to the multi-exit BERT's deployments on real-time cloud services.

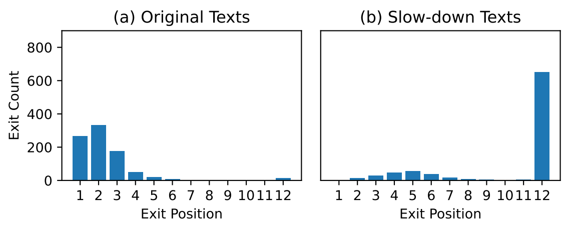


Figure 3: The exit distribution comparison on confidence-based multi-exit ALBERT with SST-2, where threshold is set as 0.95.

Table 2: Accuracy comparision between original and malicious text samples on ALBERT under both white-box (w.b.) and black-box (b.b.) scenarios.

	Without	Confidence-based criterion			Patience-based criterion		
	Early-exit	T=0.85	T=0.9	T=0.95	C=3	C=5	C=7
ALBERT + SST-2							
Origin	0.906	0.874	0.897	0.906	0.904	0.915	0.918
Ours-w.b.	-	0.650	0.644	0.647	0.635	0.635	0.646
Ours-b.b.	-	0.643	0.635	0.631	0.681	0.674	0.701
ALBERT + MPRC							
Origin	0.860	0.853	0.863	0.855	0.826	0.865	0.87
Ours-w.b.	-	0.707	0.713	0.707	0.660	0.700	0.780
Ours-b.b.	-	0.693	0.693	0.733	0.660	0.727	0.747

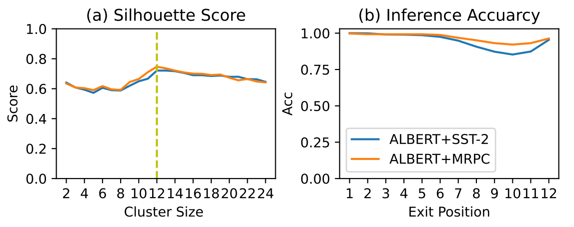


Figure 4: The performance of the time-based approximation algorithm on multi-exit ALBERT, where (a) shows the scores of K-means and (b) shows the exit position inference accuracy.

Influence on Model Utility. Further, we study whether our *SlowBERT* will cause extra damages to the model utility. As reported in Tab. 2, our attack generate malicious texts not only increase the inference costs but also degrade the classification performances of the multi-exit BERT. Specifically, the accuracy is decreased by 29% for SST-2 and 20% for MPRC on average, which calls for more attentions on our proposed attack.

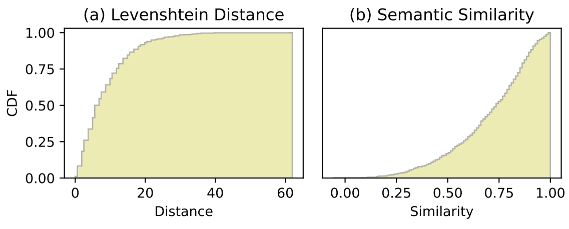


Figure 5: The text similarity distribution on confidence-based multi-exit ALBERT with SST-2, where threshold is set as 0.95.

Text Similarity. For the text similarity between the original texts and slow-down malicious texts, we first evaluate the character-level similarity using the levenshtein distance. As shown in Tab. 3, we observe that most cases only require less than 10 operations on average. Table 3 also reports that most cases generate malicious texts with high semantic similarity (i.e., 0.65-0.9 on average). An example of the similarity distribution are shown in Fig. 5, where 93% texts require less than 20 operations and 85% malicious texts have semantic similarity higher than 0.5 with the original texts. The above results indicate that our *SlowBERT* generates high quality slow-down samples to attack the multi-exit BERT. The examples of original and malicious sentences and more text similarity results can be found in Appendix B.

Table 3: The average query number and similarity between the original texts and the malicious text with multi-exit ALBERT, where w.b. denotes white-box scenario and b.b. denotes black-box scenario.

Metric	Confidence-based criterion			Patience-based criterion		
	T=0.85	T=0.9	T=0.95	C=3	C=5	C=7
ALBERT + SST-2 - w.b.						
Query Number	87.455	70.591	57.854	84.029	60.717	47.258
Levenshtein Distance	12.008	10.670	8.763	11.390	9.269	7.123
Semantic Similarity	0.627	0.662	0.703	0.657	0.695	0.757
ALBERT + MPRC - w.b.						
Query Number	110.274	79.142	66.568	138.053	94.818	75.716
Levenshtein Distance	9.800	8.141	6.568	11.104	7.913	6.401
Semantic Similarity	0.781	0.831	0.860	0.753	0.807	0.856
ALBERT + SST-2 - b.b.						
Query Number	88.769	71.960	58.935	87.345	66.311	53.981
Levenshtein Distance	11.193	9.772	8.451	9.843	8.201	7.151
Semantic Similarity	0.649	0.677	0.711	0.699	0.737	0.758
ALBERT + MPRC - b.b.						
Query Number	72.083	38.833	27.774	96.642	55.495	37.421
Levenshtein Distance	10.901	8.659	6.889	12.806	9.620	7.000
Semantic Similarity	0.804	0.834	0.865	0.791	0.834	0.880

Attack Efficiency. We also study the attack efficiency by measuring the average query number our attack made to the multi-exit BERT. As reported in Tab. 3, most cases only require less than 100 queries on average to achieve the slow-down target, which is considered to be highly efficient comparing to existing text adversarial attack (Jin et al., 2020). Finally, from Tab. 3, we observe that the text similarity and the attack efficiency is higher when the early-exit condition is more restricted, since the attack is easier.

6 Conclusion

In our work, we systematically study the potential vulnerability of the input-adaptive BERT from the perspective of large language model efficiency. We propose *SlowBERT*, which drastically increases the average inference cost of the multi-exit BERT. This new attack surface poses serious challenges to the deployments of the multi-exit BERT on real-time cloud services or local hardware-constrained edge devices. As a security problem of the large language model, our work welcomes future research to devise strong defense against our attacks.

Limitations

Apart from the effective attack performance against multi-exit BERT, we acknowledge that our work has several limitations. Firstly, we only evaluate our *SlowBERT* on the GLUE Benchmark (Wang et al., 2018a), which demonstrate the effectiveness on alphabetic languages such as English. However, for logograms (e.g., Chinese), it requires to design language-specific method to generate the corresponding substitution set to achieve the attack goal. Secondly, despite we present a new security threat against multi-exit BERT, potential defenses should be analyzed such as adversarial training (Geng et al., 2021). For the above mentioned limitations, we leave them as the future works of our proposed *SlowBERT*.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200), National Natural Science Foundation of China (61972099, U1736208, U1836210, U1836213, 62172104, 62172105, 61902374, 62102093, 62102091). Min Yang is a faculty of Shanghai Institute of Intelligent Electronics & Systems, Shanghai Insitute for Advanced Communication and Data Science, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China. Mi Zhang and Min Yang are the corresponding authors.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Steven Bohez, Pieter Simoens, Piet Demeester, and Bart Dhoedt. 2015. Distributed neural networks for internet of things: The big-little approach. In *International Internet of Things Summit*, pages 484–492. Springer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. 2017. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1039–1048.

Shijie Geng, Peng Gao, Zuohui Fu, and Yongfeng Zhang. 2021. Romebert: Robust training of multi-exit bert. *arXiv preprint arXiv:2101.09755*.

Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. 2020. Ilfo: Adversarial attack on adaptive neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14264–14273.

Mirazul Haque, Yaswanth Yadlapalli, Wei Yang, and Cong Liu. 2022. Ereba: Black-box energy testing of adaptive neural networks. *arXiv preprint arXiv:2202.06084*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. 2020. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*.

Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. 2020. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. *arXiv preprint arXiv:2002.10025*.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

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 the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Zheng Li, Yiyong Liu, Xinlei He, Ning Yu, Michael Backes, and Yang Zhang. 2022. Auditing membership leakages of multi-exit networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1917–1931.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*.

Stuart Lloyd. 1982. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ben Taylor, Vicent Sanz Marco, Willy Wolff, Yehia Elkhatib, and Zheng Wang. 2018. Adaptive deep learning model selection on embedded systems. *ACM SIGPLAN Notices*, 53(6):31–43.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. 2018b. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

A Evaluation Setting

Fisrt, we give more detailed description to the evaluation setting.

Detial of Early-exit Capacity. Early-exit capacity is a metric propose to specifically measure the efficiency performance of the input-adaptive multi-exit mechanisms. The calculate of early-exit capacity mainly consist of 3 steps. Firstly, given an exit distribution shown in Fig. 7(a), the metric gets the portion of texts that early exit at each exit position. Secondly, given the density distribution, the metric calculates the cumulative distribution curve in Fig. 7(b). Lastly, the metric calculates the size under the curve to denote the efficiency.

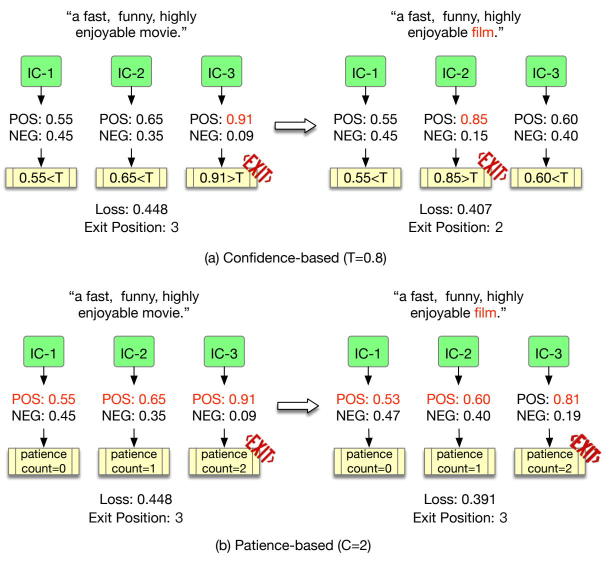


Figure 6: Baseline case study of its limited effect on multi-exit BERT with a binary classification task, where (a) denotes the undesirable case in confidence-based inference, and (b) denotes the undesirable case in patience-based inference.

Detail Setting. First, we report the parameter size of our multi-exit ALBERT is 12M, and the parameter size of our multi-exit BERT is 108M.

All our experiments are conducted on a Linux server running Ubuntu 16.04, one AMD Ryzen Threadripper 2990WX 32-core processor and one NVIDIA GTX RTX2080 GPU. The training of multi-exit BERT is conducted on random seed 47 and the construction of slow-down samples is conducted on random seed 2022.

For attacking the multi-exit BERT, under the black-box scenario, we reshape the one-dimensional time value to two-dimension for accurate clustering and use the euclidean metric in silhouette score to choose the appropriate cluster.

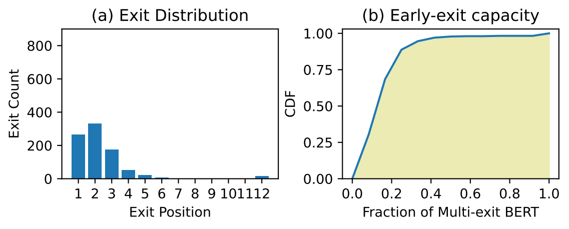


Figure 7: The detail of the early-exit capacity metirc. (a) The exit distribution. (b) The cumulative distribution curve, where we calculate the area size as the metirc score.

Baseline Analysis. Hong et al. (2020) proposed a slow-down attack against multi-exit neural networks (e.g., ResNet He et al., 2016) in vision domain. The attack specifically targets the *confidence-based* multi-exit networks by setting the attack goal as following:

$$L_{baseline} = \sum_{i=1}^{K} L_{ce}(F_i(x), \bar{y}), \tag{7}$$

where $\bar{y}$ is the uniform distribution over all class labels. The goal calculates the distance between the output of each internal classifier and the uniform distribution. By directly optimizing the images through gradient $\nabla_x L_{baseline}$, the attack generates malicious slow-down image samples, which ultimately minimize the value of $L_{baseline}$ to 0.

Considering the typical search-based adversarial text generation algorithm (Jin et al., 2020; Li et al., 2018), we integrates the baseline's attack goal with our rank-and-substitute algorithm, where the rank-and-substitute algorithm aims to search for the appropriate word in the direction of decreasing the value of Eq. 7. However, we provide several cases in Fig. 6 to show that the baseline's attack goal is not suitable for the slow-down attack against multi-exit BERT. For *confidence-based* criterion, as shown in Fig. 6(a), since the loss is calculated by averaging the result of each internal classifier, the generated text may exit earlier than the previous ones even with lower loss. For *patience-based* criterion, as shown in Fig. 6(b), the dropping of the loss may only cause the output of the internal classifiers closer to the uniform distribution, but the prediction labels remains unchanged.

Compared with our attack goal in Eq. 6, we directly incorporate the exit position into the target goal, which gives a clearer goal in the search process. Since the typical search-based algorithm can't directly optimize text sample in the continu-

Table 4: Accuracy comparision between original and malicious text samples on BERT under both white-box (w.b.) and black-box (b.b.) scenarios.

	Without	Confidence-based criterion			Patience-based criterion		
	Early-exit	T=0.85	T=0.9	T=0.95	C=3	C=5	C=7
BERT + SST-2							
Origin	0.915	0853	0.880	0.896	0.884	0904	0.912
Ours-w.b.	-	0.689	0.687	0.671	0.683	0.674	0.685
Ours-b.b.	-	0.695	0.710	0.704	0.712	0.702	0.714
BERT + MRPC							
Origin	0.828	0.814	0.824	0.828	0.701	0.806	0.816
Ours-w.b.	-	0.586	0.593	0.673	0.353	0.533	0.533
Ours-b.b.	-	0.560	0.613	0.660	0.387	0.593	0.620

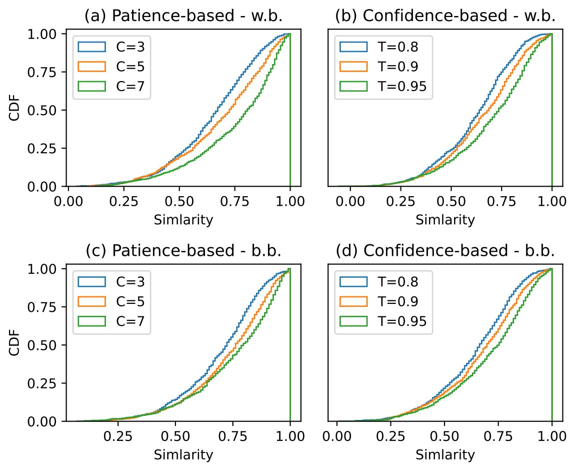


Figure 8: The semantic similarity on multi-exit ALBERT with SST-2, where (a)-(b) shows the similarity under the white-box scenario and (c)-(d) shows the similarity under the black-box scenario.

ous space, we argue that our proposed *SlowBERT* attack is more suitable and effective.

B Evaluation Result

In this section, we provide more evaluation results to demonstrate the effectiveness of our *SlowBERT*.

Exit Distribution. We first report the exit distributions of the multi-exit ALBERT with SST-2 under different multi-exit settings. As we can see from Fig. 10, our attack can generate malicious texts that successfully delay the exit positions across every cases, which ultimately leads to the increase of the inference cost. We can also observe that more malicious texts will exit at the last internal classifier as exit condition is gradually more restricted (i.e., the increase of the threshold T and the patience count C).

Influence on Model Utility. We presents more results related to the influence of the malicious texts on the model utility of the multi-exit BERT. As reported in Tab. 4, the malicious texts targeted the efficiency of the multi-exit BERT also degrade the classification accuracy on both datasets. Specifically, the accuracy drop by 25% and 30% for SST-2 and MRPC on average.

Table 5: The average query number and similarity between the original texts and the malicious text with multi-exit BERT, where w.b. denotes white-box scenario and b.b. denotes black-box scenario.

Metric	Confidence-based criterion			Patience-based criterion		
	T=0.85	T=0.9	T=0.95	C=3	C=5	C=7
BERT + SST-2 - w.b.						
Query Number	66.344	48.330	35.401	65.670	35.429	23.291
Levenshtein Distance	11.062	9.688	8.390	10.850	8.178	6.259
Semantic Similarity	0.637	0.681	0.717	0.662	0.724	0.782
BERT + MRPC - w.b.						
Query Number	77.833	64.951	55.701	145.804	105.319	59.833
Levenshtein Distance	8.282	6.867	5.381	18.743	9.780	6.052
Semantic Similarity	0.845	0.871	0.892	0.518	0.826	0.875
BERT + SST-2 - b.b.						
Query Number	67.782	49.142	34.589	66.544	39.544	25.775
Levenshtein Distance	10.103	9.221	7.261	9.413	7.107	5.847
Semantic Similarity	0.660	0.694	0.740	0.685	0.750	0.795
BERT + MRPC - b.b.						
Query Number	40.495	28.113	16.054	102.480	66.363	27.025
Levenshtein Distance	7.376	6.193	5.468	14.457	8.002	5.574
Semantic Similarity	0.860	0.880	0.888	0.629	0.842	0.889

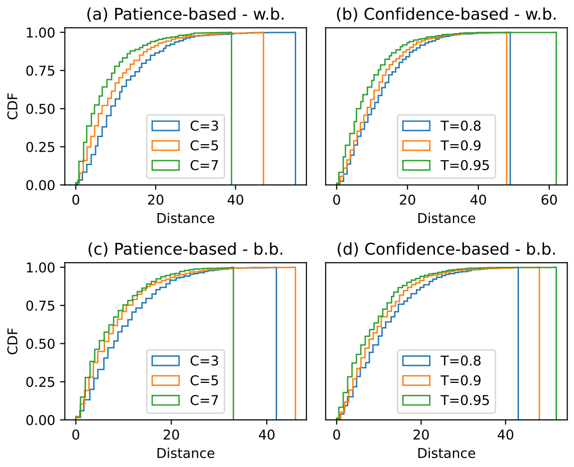


Figure 9: The levenshtein distance on multi-exit ALBERT with SST-2, where (a)-(b) shows the distance under the white-box scenario and (c)-(d) shows the distance under the black-box scenario.

Text Similarity. As to the text similarity, Table 5 first report the results on multi-exit BERT. We make the following finding similar to the ones made on multi-exit ALBERT. Most cases only requires less than 10 operations on average to modified the original texts to the malicious text. And for semantic similarity, most cases have similarity higher than 0.6 except for one case with 0.518.

To give a comprehensive view of the text similarity, we plot the distribution of the levenshitein

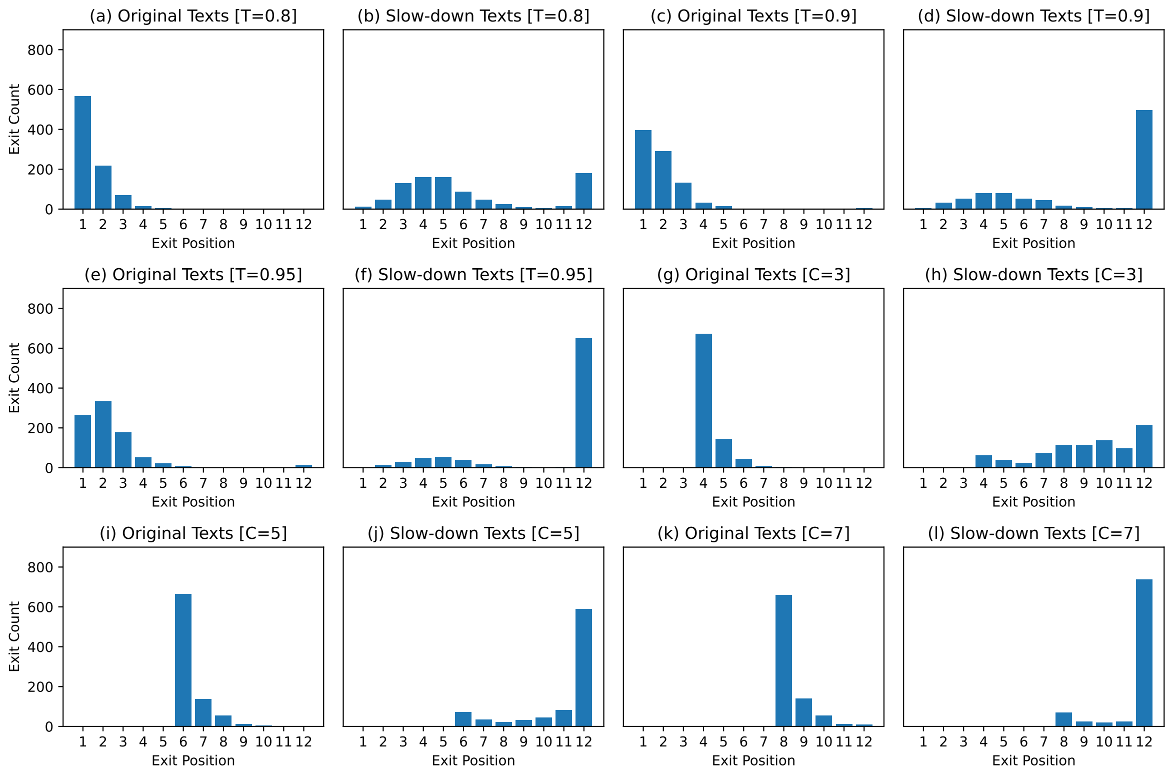


Figure 10: The exit distribution on multi-exit ALBERT with SST-2, , where (a)-(f) shows the distributions of confidence-based multi-exit criterion and (g)-(l) shows the distributions of patience-based multi-exit criterion.

Table 6: Examples of original and slow-down sentences from SST-2 and MRPC on multi-exit ALBERT.

	SST-2+ALBERT	
Origin	Exit: 3	although **laced** with **humor** and a few fanciful touches , the **film** is a **refreshingly** serious **look** at young **women**
Malicious	Exit: 12	although **laecd** with **humod** and a few fanciful touches , the **f ilm** is a **refrshingly** serious **expression** at young **wonen**.
Origin	Exit: 4	for the most part, director anne-sophie birot's first feature is a **sensitive**, extraordinarily **well-acted** drama.
Malicious	Exit: 12	for the most part, director anne-sophie birot's first feature is a **sesitive**, extraordinarily **dell-acted** drama .
	MRPC+ALBERT	
Origin	Exit: 4	"**Jeremy** 's a good guy, "Barber said, adding: "Jeremy is **living** the dream life of the New York **athlete**.
Malicious	Exit: 12	"**Jreemy** ' s a good guy, "Barber aforesaid , adding : "Jeremy is **livng** the dream life of the New York **jock**.
Origin	Exit: 4	University of Michigan President Mary Sue Coleman said in a statement on the university 's Web site , " Our fundamental **values** haven't **changed**.
Malicious	Exit: 11	University of Michigan President Mary Sue Coleman said in a statement on the university 's Web site , " Our fundamental **valus** haven't **cahnged**.

distance and the semantic similarity respectively obn multi-exit ALBERT with SST-2 in Fig. 9 and Fig. 8. We demonstrate the text similarity for different multi-exit settings under both white-box and black-box scenarios. The findings are consistent with those in Sec. 5.

We can observe that most malicious texts require less than 20 operations during the attack. And most malicious texts are highly semantic similar to the original texts. In terms of the influence of different exit conditions, we find that harder exit conditions tend to make our attack generate malicious texts with higher text similarity (i.e. lower levenshtein distance and higher semantic similarity).

Query Number. Furthermore, We report the attack efficiency on multi-exit BERT from the perspective of the query number. As we can see from Tab. 5, most cases require less than 80 times of queries to achieve the attack goal. From Tab. 5, we can also conclude that the query number is negatively linear correlated with the strictness of the exit condition.

Example of the texts. Finally, the examples of the original and malicious texts are presented in Tab. 6. This shows that the malicious texts only require limited modifications and remain semantic similar to the original ones.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
in section7

☑ A2. Did you discuss any potential risks of your work?
we discuss the threat model of our SlowBERT attack against the practical deployments in Section1 and Section3.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
in section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

In section 5 "detail settings"

☑ B1. Did you cite the creators of artifacts you used?
In section 5 "detail settings"

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
In section 5 "detail settings"

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☑ Did you run computational experiments?

In section5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report the numbe of parameters in appendix. our attack was performed on cpu.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
In section5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We report it in appendix.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
We used the pretrained bert/albert provided in hugging face's transformer, which we discussed in section5.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.