Understanding, Uncovering, and Mitigating the Causes of Inference Slowdown for Language Models

Kamala Varma*, Arda Numanoğlu†, Yigitcan Kaya‡, Tudor Dumitraș*
*Department of Computer Science, University of Maryland, College Park
{kvarma, tudor}@umd.edu
†Department of Computer Engineering, Middle East Technical University, Ankara
arda.numanoglu@metu.edu.tr
‡Department of Computer Science, University of California, Santa Barbara
yigitcan@ucsb.edu

***Abstract*—Dynamic neural networks (DyNNs) have shown promise for alleviating the high computational costs of pre-trained language models (PLMs), such as BERT and GPT. Emerging *slowdown attacks* have shown to inhibit the ability of DyNNs to omit computation, e.g., by skipping layers that are deemed unnecessary. As a result, these attacks can cause significant delays in inference speed for DyNNs and may erase their cost savings altogether. Most research in slowdown attacks has been in the image domain, despite the ever-growing computational costs—and relevance of DyNNs—in the language domain. Unfortunately, it is still not understood what language artifacts trigger extra processing in a PLM or what causes this behavior. We aim to fill this gap through an empirical exploration of the slowdown effect on language models. Specifically, we uncover a crucial difference between the slowdown effect in the image and language domains, illuminate the efficacy of pre-existing and novel techniques for causing slowdown, and report circumstances where slowdown does not occur. Building on these observations, we propose the first approach for mitigating the slowdown effect. Our results suggest that slowdown attacks can provide new insights that can inform the development of more efficient PLMs.**

***Index Terms*—adversarial machine learning, pre-trained language models, efficient machine learning**

I. Introduction

PLMs such as GPT-4, Bard, or BERT, are large models pre-trained on massive corpora of text with the intention of learning a general understanding of language. Bolstered by this base understanding, PLMs can be fine-tuned on task-specific datasets to perform a wide range of language tasks. PLMs have high parameter counts, ranging from 110 million to 345 million for BERT and exceeding 1.5 trillion for commercial models like GPT-4 or Bard, suggesting high computational expense is inherent to their utility [1]. To reduce the cost of PLM inferences, researchers have proposed dynamic neural networks (DyNNs), which adapt the processing performed during the forward pass to individual input values, as some inputs require less computation than others [2]. For example, in multi-exit models (MEMs), during inference, textual inputs can exit the model at various points that precede the final output layer [3]–[5].

However, MEMs are vulnerable to the recently-introduced concept of a slowdown attack, which was originally proposed against vision models in [6] and has now found a sole, initial implementation against language models in [7]. Slowdown attacks are designed to delay the exits of inputs and have the potential to completely negate all of the efficiency benefits of MEMs. There are no known defenses against this threat; in vision models, adversarial training was shown to be ineffective against slowdown attacks [6]. The research focus on developing effective and powerful attacks has left missing a deeper understanding of slowdowns, such as their sources and implications. The goal of this paper is to provide a systematic empirical exploration of these questions in the context of large language models, where the cost savings promised by MEMs are critical. A better understanding of the slowdown effect can ultimately inform the development of language models that are both more efficient and more robust in their efficiency.

We focus our exploration on studying an exemplar PLM, BERT (Bidirectional Encoder Representations from Transformers) [8], commonly used as a standard for NLP research, and whose descendants achieve state-of-the-art performance on natural language understanding benchmarks. In the first part of the paper, Sec. IV, we explore a range of approaches to understand the source of the slowdown effect. First, we investigate if the PLM's need for additional processing can be explained by simple metrics related to the ease of comprehension of the input sentences. We empirically reject three intuitive hypotheses about the origins of slowdown: increased distances among sentences and words in BERT embedding space, decreased cloze probabilities of tokens, and the use of paraphrased text. Then, we analyze the efficacy of two methods that do inflict slowdown. The first is the SlowBERT slowdown attack [7] that has proven to be highly effective, but in nuanced ways that we will uncover. The second is an inadvertent slowdown effect that we are the first to identify as being inherent to pre-existing text misclassification attacks, which contrasts a prior observation on image misclassification attacks [6]. After offering explanations for this contrast, which

we illustrate through experimental results, we introduce a novel third method for causing slowdown that can additionally cause inference speedup. This involves the development of a ML model that, based on a new task we propose, learns a unique understanding of what distinguishes texts that cause slowdown from ones that do not. We use our insights from the first part of the paper in conjunction with an additional set of empirical results to propose an attack mitigation strategy. To the best of our knowledge, this represents the first defense against slowdown attacks against MEMs. This method, a quicker, relaxed version of adversarial training that we refer to as *expedited adversarial training*, we will show to be effective in mitigating slowdown effects that are imposed both intentionally and unintentionally through a diverse set of strategies. Additionally, this can provide inference *speedup* for benign text. In summary, we provide the following five central contributions:

1) We evidence the fact that increasing distances between words and sentences in BERT embedding space, lowering a sentence's aggregate cloze probability, or paraphrasing sentences does not exhibit association with the slowdown effect.
2) We show that existing text misclassification attacks cause slowdown as a side effect—unlike in the vision domain where this does not occur—and we investigate the sources of this effect.
3) We analyze the sole slowdown attack that has been proposed in the text domain, SlowBERT, reporting a discrepancy in the effectiveness of its modification methods and highlighting the increased slowdown potential that comes from SlowBERT's word-selection strategy.
4) We demonstrate the novel idea that a machine learning model can learn a unique understanding of the distinction between texts that amount to earlier or later exiting. We use this learned model to drive a novel slowdown attack generation process.
5) We demonstrate and offer an explanation behind the effectiveness of a flexible expedited adversarial training process in mitigating slowdown effects that come from multiple distinct sources.
6) We provide a code library that includes implementations of all the attacks and defenses that our work analyzes.[1]

II. Related Work

a) Conventional Attacks against Language Models: The goal of most pre-existing adversarial attacks against text classification tasks is to cause misclassification. These attacks find places in original input texts to replace original words with new words (often synonyms) or to append/delete/swap words or characters, with all modification choices being guided by greedy or beam searches, genetic algorithms, and various patterns in model gradients or token embeddings [9]–[21].

[1]https://github.com/KMVarma/SwiftBERT

b) Input-Adaptive Networks: Broadly, dynamic neural networks (DyNNs) are any type of neural network that can adapt its structure, computations, and parameters during inference in a way that is tailored to a specific input and offers some sort of advantage. In our work, we focus on multi-exit models (MEMs), which are models that increase inference efficiency by allowing inputs to exit at layers preceding the output layer. Essentially, [22] found that inputs can often be accurately predicted before reaching the output layer, with some inputs reaching this point earlier than others, so various MEMs have been proposed to save on any extra computation deemed unnecessary for a particular input in both the image [22], [23] and text domains [3]–[5]. Specifically, inputs can exit through internal classifiers (ICs) upon meeting some exiting criteria. ICs were first introduced to vision models in [22] and they take various approaches to, for a single input, mapping a layer's hidden state to a prediction value. Finally, in the era of trillion-parameter language models, input-adaptiveness has seen recent practical deployment. Related ideas such as speculative decoding [24] or mixture-of-expert models [25] have been reported to be implemented in GPT-4 [26] for adaptively reducing its massive computational load. This implies that there may be an imminent real-world threat stemming from slowdown attacks, which brings forth the need for a deeper exploration.

c) Slowdown Attacks: The concept of a slowdown attack was first introduced to the vision domain in the work by [6] that proposed the DeepSloth attack. DeepSloth attacks image classification tasks, aiming to cause slowdown on MEMs, or in other words, to perturb inputs in such a way that requires them to exit through later ICs and therefore incur unnecessary additional computation. It accomplishes this through optimizing an attack perturbation over an objective function that encourages each IC to predict class confidence scores corresponding to a uniform distribution over all possible class labels. Other work in the image space also explores the idea of crafting an attack with the intention of slowing down a model or increasing a model's latency in some fashion [27]–[33]. [34] was published concurrently with the DeepSloth paper and it introduced sponge examples to the text and image domains, which are inputs that were designed with a novel threat vector that aims to maximize latency and energy consumption for neural networks (they have also been proposed in an imperceptible form in [35]). To clarify, this is different from the DeepSloth-proposed slowdown attack threat that specifically aims to cause delayed exit in MEMs and is the threat model that our work is considering. [7] is the first work, and to our knowledge the only work, to explore the slowdown attack in the context of language. Their proposed SlowBERT attack considers a variety of modification types while searching for the best modification to cause slowdown and the search is guided by the strategy used in by the DeepSloth of pushing a sample's IC confidence scores towards a uniform distribution. However, they have not presented a systematical treatment of their attack's results based on the type of modification. To fill this gap, in Sec. IV-B2, we carefully investigate how effective

each type is in causing slowdown.

d) Increasing the Adversarial Robustness or Efficiency of Language Models: Many works have explored a variety of approaches to using adversarial training and similar methods to increase the robustness of language models against adversarial attacks [9], [15], [36]–[40]. There has also been research done on language models to learn and exploit relationships surrounding the inference speed, latency, and energy consumption of language models in order to motivate solutions for increasing and preserving model efficiency [41]–[43]. However, in Sec. V, we present what is, to our knowledge, the first analysis of the efficacy that a modified version of adversarial training has in terms of mitigating slowdown attacks while additionally provoking inference speedup for MEMs in the text domain.

III. Preliminaries

1) Determining Exit Points for Multi-Exit Models: We consider MEMs that use the same IC definition that is formalized in [4]. Every layer of a MEM will be coupled with its own IC. We can use these ICs to derive confidence scores at every layer. Essentially, as an input passes through each layer of the network, the layer will output a new hidden state. This hidden state is then passed through the layer's IC, and this IC has been trained to map the layer's hidden state to output logits representing the unnormalized probablities associated with each possible class labels. These logits are then utilized as a confidence score. Specifically, for our work, we consider confidence scores as the softmax score associated with the output logits. In other words, we apply a softmax function to the logits and are left with confidence scores that represent the normalized probabilities associated with each class. Prior work mainly utilizes some form of the following two exiting criteria: confidence-based and patience-based criteria. According to confidence-based exiting criteria, an input can exit at some layer, layer P, if that layer's IC predicts a confidence score above some threshold, τ, for a single class that then becomes the final predicted class. With patience-based exiting criteria, an input exits at layer P if it marks the first point at which p consecutive ICs have agreed on the same class prediction (the class receiving the highest confidence score) [4]. For both criteria types, if an input never meets the criteria required for an early exit, then it will default to exiting at the output layer.

2) Identifying Tokens Most Vulnerable to Attacks: A common approach that text misclassification attacks will take to identify the best places to modify the text is to use some sort of heuristic to rank an input text's tokens based on how good of a candidate each one is for modification. As part of an analysis in Paragraph IV-B2b of Sec. IV-B2, as a heuristic, we will measure a simple version of saliency, S, for some token t_i, defined similarly to the way that an importance score is defined in [19]:

$$S = F^c(x) - F^c(x_{t_i}), \tag{1}$$

where $F^c(\cdot)$ is the model's final confidence score corresponding to the correct class label, x is the original text sample that token t_i is in, and x_{t_i} is the sample with t_i removed. The most salient token (the one with the highest S value) in an input text is the one whose modification would have the most potential to move a model's final class confidence scores away from those produced before the modification. We will compare the utility of S with that of exit status H, which is the measure of efficiency relevance used in the SlowBERT attack to identify the most slowdown-vulnerable words that should be modified (SlowBERT is designed to modify words as opposed to tokens). It is formalized as:

$$H = \alpha \cdot P - L_{ce}(F_P(x), \bar{y}), \tag{2}$$

where x is a text sample (input), P is the exit layer taken by x, L_{ce} is the cross-entropy loss, F_P is the output of the internal classifier at exit layer P, and $\bar{y}$, the target for the internal classifier's output which is a uniform distribution across all possible labels. α is a tunable, predetermined constant that should be set high enough to allow H to favor a word's impact on the exit position.

A. Threat Model

We study a threat model similar to that defined in [7].

a) Adversary's Goal: We consider an adversary whose goal is to craft a slowdown attack by modifying an input text such that the modified text exits later when passing through a BERT-based MEM than the original text. This delay in inference will impose additional costs on the model deployer that could amount to detrimental consequences. For example, if slowdown attacks cause a node in a distributed system to incur longer response times, the system may lose information from the node by mistakenly declaring it dead. In such attacks, it has been shown that the addition additional cost the victim suffers from due to the attack can surpass the adversary's crafting cost; making the threat practical [6].

b) Adversary's Capabilities: The adversary has no restrictions on the way that they modify input texts. Such capability is available to adversaries in modern ML practice as oftentimes models are fed untrustworthy inputs scraped from online sources, e.g., social media posts, that cannot be realistically vetted. These modifications can include adding, deleting, or swapping any number of words, tokens, or characters.

c) Adversary's Knowledge: We consider adversaries that have any of three access levels, which we define as follows:

- **White-box:** The adversary has knowledge of the target model's IC count and the early-exit criteria in use, the model's parameters, and the exit positions and IC outputs corresponding to the target input samples. This is used in the version of the SlowBERT attack that we experiment with.
- **Grey-box:** The adversary only has knowledge of the layers at which each target input exits the target model. We use this for the novel slowdown attack that we introduce in Sec. IV-C3.
- **Black-box:** The adversary requires nothing beyond access to the target input samples. This is essentially the grey-box scenario, but the adversary can use exit layers

computed from a surrogate pre-trained (but not necessarily fine-tuned) BERT model to estimate the exit layers associated with a fine-tuned pre-trained BERT model that the adversary is targeting. Therefore, this can also be used for our novel slowdown attack in Sec. IV-C3.

B. Experiment Setup

Throughout this paper, we present a series of empirical observations, each involving experiments whose setups include the following components. We consistently use 12-layer BERT-based MEMs, where during inference, an input can exit the model early through ICs positioned at every layer. We focus on observing slowdown patterns surrounding patience-based exiting criteria because, compared to other criteria, it has proven to yield the best accuracy-speed trade-off and to be the most robust. With the patience-based criteria, we mostly consider $p = 5$ or $p = 6$. We find that, in most settings, these values tend to be the smallest for which accuracy is not significantly compromised (smaller values are favorable because they offer the most potential to reduce inference costs). We source a pre-trained BERT model from Hugging Face [44] and fine-tune it on two downstream classification tasks from the popular General Language Understanding Evaluation (GLUE) benchmark [45]. The Stanford Sentiment Treebank (SST-2) [46] task assigns a binary sentiment classification to individual sentences and the Microsoft Research Paraphrase Corpus (MRPC) [47] task is a binary classification task for identifying the semantic equivalence of a pair of input sentences. In Appendix E, we provide results associated with a third GLUE benchmark, the Multi-Genre Natural Language Inference (MNLI) [48] task, which assigns one of three labels based on a premise sentence's entailment of a hypothesis sentence. Any time we use synonyms in our work, we find these using WordNet [49].

IV. Demystifying the Slowdown Effect on Pre-Trained Language Models

In this first part of the paper, we undertake a methodical exploration of the slowdown effect in MEMs in an attempt to better understand what causes slowdown in the text domain and why. We present three distinct phases of our exploration. First, in Sec. IV-A, we explore hypotheses based on our intuitions about the patterns that we expect to align with the slowdown effect, but interestingly, our findings do not support these hypotheses. This leads us to Sec. IV-B, where we analyze existing attack strategies that either intentionally or unintentionally cause slowdown in order to better understand where the slowdown effect is coming from. Finally, in Sec. IV-C, we present a third approach to understanding the slowdown effect, where we make the novel observation that a machine learning (ML) model can learn a unique understanding of the distinction between earlier and later-exiting text. We then show how this model can be used in a novel slowdown attack generation method.

A. Places in Which Evidence of Slowdown is Unexpectedly Absent

1) Increased Distance Between Token and Sentence Embeddings is not Associated with More Pronounced Slowdown: We initially expected the slowdown effect to somehow be evident in BERT embedding space. Specifically, we hypothesized that, for some sentence that has had words replaced in a way that leads to slowdown, the distance between the replacement words' embeddings and the sentence embedding would be greater than the distance between the original words' embeddings and the sentence embedding. This supposition was rooted in the idea that an increased distance from the sentence embedding would imply that the replacement words do not have contextual relevance to the rest of the sentence that is as high as the relevance of the original words. In other words, after the replacement, the consistency among the words' contexts is decreased, hence making the sentence as a whole more confusing for BERT to reason about. To test this theory, we extracted word embeddings associated with SlowBERT attacks on the SST-2 dataset and compared the distances between the original words and the full sentence embedding against the replacement words and the full sentence embedding. Interestingly, our analysis revealed that, using patience-based exiting criteria with $p = 6$, only **41.5%** of attacked sentences exhibited greater euclidean distances in replacement word embeddings from the sentence embedding and **54.1%** experienced increased cosine distances. This observation leads us to conclude that increased distance between a sentence's words embeddings to the overall sentence embedding is not indicative of a slowdown effect.

2) Lower Cloze Probability Does Not Necessarily Indicate Greater Slowdown: Cloze probability is a widely recognized concept in psycholinguistics, serving as a measure of a word's predictability within a specific context. In their work, [50] addresses the challenges associated with measuring cloze probabilities through direct surveys of human subjects, motivating their use of GPT-2 to generate reliable estimates of cloze probabilities. Our initial hypothesis was that a greater slowdown effect may be associated with lower cloze probabilities. This is because we expect BERT to be able to more easily comprehend sentences with patterns that best align with the texts it was trained on. These texts were online-sourced, human-written, and would therefore likely have high GPT-2-estimated cloze probabilities. To investigate the potential link between slowdown and lower cloze probabilities, we crafted an attack that modifies the SlowBERT implementation (original implementation summarized in Sec. IV-B2). In this modified attack, during the process of selecting a single modification from the set of candidate modifications, we chose the modification that resulted in the lowest aggregate cloze probability for the sentence. Aggregate cloze probabilities were computed by averaging the sentence's GPT-2-computed individual word cloze probabilities. This attack was carried out using the SST-2 dataset and a fine-tuned BERT MEM that use patience-based exiting criteria with $p = 5$. The resulting

attacked sentences produced an average exit layer of **8.425**, or a 16.90% increase from the original average exit of **7.207**. In contrast, using the original SlowBERT implementation led to an average exit layer of **11.940**, which represents a 65.67% increase. It is also worth noting that later, in Table III of Sec. IV-B1, we will present results associated with an MRPC dataset wherein randomly-selected words were replaced with randomly-selected synonyms, which leads to a 13.26% increase in average exit layer. In fact, throughout this paper, we will provide results and explanations that can ultimately support the idea that a small slowdown effect can easily be instantiated by almost any means of modifying an input text. Hence, we conclude that the slowdown effect caused by cloze-informed text modifications is not great enough to indicate a notable, direct association between cloze probabilities and ability that texts have to exit early in a MEM.

3) Paraphrasing Models Do Not Prove Effective in Inducing Slowdown: Prior work has demonstrated the potential that paraphrasing models have to function as adversaries in the language domain [51]. Consequently, we hypothesized that paraphrasing texts using a paraphrasing model may induce a slowdown effect. To investigate this, we conducted experiments using two publicly-available, state-of-the-art, distinctly-structured paraphrasing models, Pegasus [52] and Parrot [53], to rephrase sentences from the SST-2 dataset. We randomly selected 100 sentences and paraphrased them with each model, but each model's paraphrasing only caused negligible change in average exit layer (3.108% change for Pegasus and 4.234% for Parrot), thus implying that paraphrasing text does not cause slowdown.

B. Explaining the Efficacy of Existing Attacks That Cause Slowdown

1) Investigating the Inadvertent Slowdown Effect Caused by Misclassification Attacks: [6] observed that with vision tasks, adversarial attacks that intend to cause misclassification will not cause slowdown. In contrast, we observe that adversarial attacks aiming to cause misclassification for language tasks do inadvertently cause a significant slowdown effect. In Table I, using the SST-2 and MRPC datasets (results for MNLI are in Table VII in Appendix E), we report slowdown effects caused by TextFooler, PWWS, and BAE attacks, whose implementations we source from the TextAttack library [54]. These attacks range between causing **8.96%** and **24.72%** increases in average exit layer. We posit that this occurrence is due to the fact that forcing the output layer towards an incorrect label will naturally cause patterns in the class confidence scores of the hidden layers that indicate model confusion or indecision and can therefore be linked to slowdown. To reiterate, the confidence scores that we refer to in our work are computed as explained in Sec. III-1 (the softmax scores associated with the logits that are output buy ICs that are coupled with every MEM layer and are trained to map the layer's output hidden state to a classification prediction).

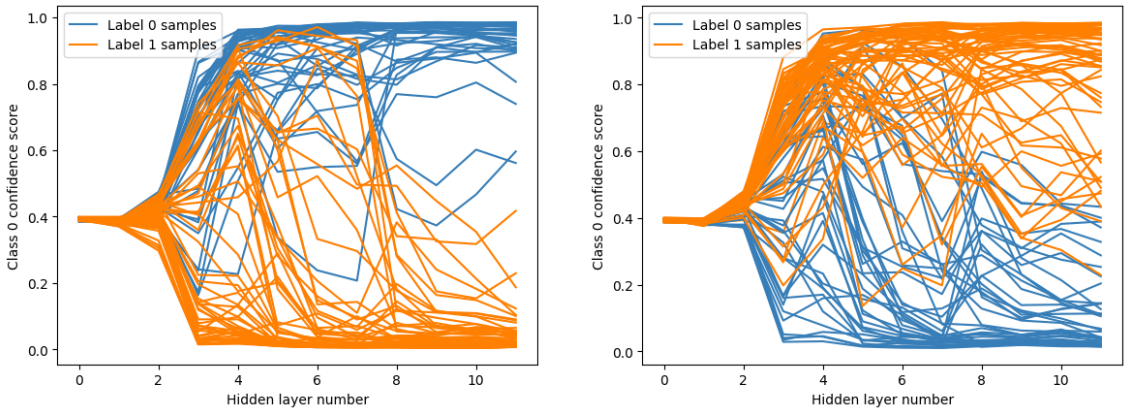
One way we observe this confusion is through confidence scores that center around a uniform distribution across all possible class labels, which [6] and [7] have shown provokes slowdown. Some misclassification attacks stop their search for effective modifications on a text as soon as a modification causes the output label to become incorrect, which in practice, is often a point at which the confidence score for this incorrect class just barely exceeds $1/n$ for an n-class classification task. Moreover, attacking text is generally difficult, more difficult than attacking images, because the attack generation is highly constrained by requirements such as preserving semantics and logical grammar. Thus, even near-optimal modification choices do not often result in high confidence towards an incorrect classification. It is not normal to see a trajectory where the confidence scores are consistently favoring a single class until suddenly ending near $1/n$ (confusion) at the output layer, so if the model is confused at the output layer, then it is likely too confused in prior layers to achieve the consistency that is required to satisfy patience-based exiting criteria. Fig. 1 exemplifies these aforementioned phenomena because, compared to before the attack, after a successful TextFooler attack, the class confidence scores corresponding to 100 SST-2 samples are leaning more towards the incorrect classification, but are also generally more centered around the .5 mark. Fig. 8b in Appendix B also illustrates this pattern of consistent model confusion.

Even when we see confidence scores more highly favoring an incorrect class at the output layer, we often observe model indecision, with confidence scores initially favoring the correct class. This implies that there is a point of switching predictions that breaks the consistency needed to satisfy patience-based exiting criteria. Fig. 8 in Appendix B separates samples according to how certain the post-attack confidence scores are in order to elucidate this pattern. Basically, the reason we see inadvertent slowdown with text misclassification attacks is because of a variety of behaviors that we see in layerwise confidence scores, which are different from what we typically see with images. On one hand, with images, misclassification attacks are known to exhibit adverse behaviors mainly at or near the output layer of a model. However, unique to language models is the fact that the adverse effects of attacks are evident throughout middle layers of a model. Works such as [55] and [56] suggest that this is because different high-level characteristics of a text input, such as syntax, semantics, and task-specific attributes, are processed at different layers within a model. An attack can directly modify any of these characteristics and a single modification can likely affect multiple characteristics, thus it makes sense the attack can take effect throughout the full span of the model as opposed to adhering towards the output layer.

2) Understanding Where the Success of the SlowBERT Attack Comes From: In this section, our objective is to analyze SlowBERT's impact on model slowdown to gain a deeper understanding of the attack's effectiveness. Once an input sentence's words are ranked based on the exit status H, the attack proceeds through an iterative modification process that considers five distinct modification methods: *Insert* (Inserting a space), *Delete* (Deleting a random character), *Swap* (Swapping

TABLE I: Comparing the average exit layer taken by 100 SST-2 or MRPC samples before and after the execution of three pre-existing misclassification attacks (TextFooler [19], probability weighted word saliency (PWWS) [17], and BERT-based Adversarial Examples (BAE) [11]) and the SlowBERT attack [7] (whose intention is to cause delayed exit) in order to demonstrate the inadvertent slowdown effect that misclassification attacks cause. Each of the 100 samples were selected from a task's test dataset based whether the specific attack could be successfully generated from the sample input, therefore each attack uses a slightly different set of 100 samples. Patience-based exiting criteria was used, with $p = 5$.

Task	TextFooler avg. exits			PWWS avg. exits			BAE avg. exits			SlowBERT avg. exits		
	Before	After	% change	Before	After	% change	Before	After	% change	Before	After	% change
SST-2	8.59	9.36	8.96	8.20	9.32	13.66	8.05	8.90	10.56	8.37	11.47	37.04
MRPC	7.20	8.98	24.72	7.49	8.60	14.82	7.84	8.13	3.77	7.43	10.39	39.84



(a) Before attack.

(b) After attack.

Fig. 1: Confidence scores associated with class 0 for 100 SST-2 samples (one line per sample, color-coded according to correct class label) at each layer. These values are compared before (a) and after (b) a successful TextFooler attack in order to illustrate patterns of model confusion that likely contribute to the inadvertent slowdown effect that misclassification attacks cause.

random two adjacent characters), *Sub-C* (Replacing characters with visually similar characters), and *Sub-W*. While the first four methods induce character-level modifications, Sub-W stands apart as it replaces a word with its nearest synonym in the word embedding space. The algorithm picks the best modification among them by leveraging the exit status H and continues modifying the words until the attacked text achieves the last exit layer.

a) Impact of Individual Modification Types on Slowdown: While the SlowBERT attack has proven highly successful in causing slowdown, the paper in which it was introduced does not present separate evaluation results for each individual word modification method that the attack uses. However, we observe a noteworthy variation in success across the different modification methods, which we illustrate through three different experiments. In our first experiment, we performed the SlowBERT attack five times for each modification method, with each attack involving a single method while excluding the others. We used the SST-2 dataset and a patience-based exiting criterion with $p = 6$ during this phase. Our results that we depict in Fig. 2 indicate that among the five modification methods, Sub-W exhibited the least impact on model slowdown, with an average exit layer of **10.11**. The Insert method showed the highest potential to induce slowdown compared to other character modification methods, with an average exit layer of **11.8076**. For context, the baseline mean exiting layer of the SST-2 dataset without modifications was **8.3094**. In Appendix A, we include results associated with a second experiment in which we vary the allowed number of modifications that the attack can make. This is to see how the added constraint may affect the efficacy of the different modification types. We consistently find that Insert is again the most successful and that character-level modifications have higher potential than Sub-W to cause slowdown.

Because inserting, deleting, and swapping characters introduce new tokens and lengthen the sequence, while synonym replacement causes a comparatively smaller increase in token count, we question whether the rise in token count drives the slowdown, independently of the modification performed. For our third experiment, we conducted the SlowBERT attack differently. Instead of choosing the substitution method with the highest exit status H, we opted for the modification that resulted in the most significant increase in tokens. Additionally, we restricted the number of allowable modifications to four, aiming to reduce the likelihood of randomly discovering an effective approach. We assessed the effectiveness of this

modified attack against the original SlowBERT, limiting the allowable modifications to four. Based on the findings, while the original SlowBERT achieves the mean exiting layer of **11.410**, the modified attack targeting the highest token number achieves an average exiting layer of **9.612**. The results of this experiment indicate that simply targeting the highest token increase does not guarantee a significant slowdown. This suggests that the superiority of character-level modifications over synonym replacement is not solely due to the rise in token count.

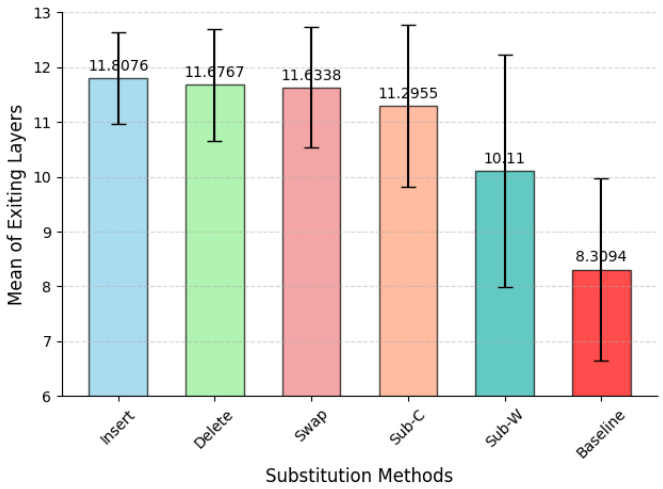


Fig. 2: Average exit layer resulting from the use of five different versions of the SlowBERT attack executed on SST-2 data, with each version solely able to use one modification type (listed as the substitution method on the x-axis), using patience-based exiting criteria with $p = 6$, and comparing against a baseline average exit layer computed on benign SST-2 data. Error bars represent standard deviation. The first four methods (the character-level modifications) generally cause more slowdown than Sub-W (a word-level modification), with Insert being the most successful.

b) Illustrating the Utility of the Exit Status H: Now, we will explain the results of an experiment comparing saliency (Equation 1) and exit status H (Equation 2), primarily to illustrate the fact that the exit status H maximally exploits words' potential to cause slowdown when modified. We take 100 SST-2 samples, delete various numbers of their *least* vulnerable words according to saliency or exit status H, and evaluate them on a fine-tuned BERT-based MEM that uses patience-based exiting criteria with $p = 5$. As evidenced in the plot in Fig. 3 (and in Table V in Appendix C, we include the associated values), we find that deleting the words from a text sample that are assumed to have the least potential to either cause the sample to be misclassified or delay the sample's inference, results in inference speedup without significantly compromising accuracy. Using saliency to guide the word deletion is slightly more favorable to prediction accuracy, while exit status H leads to more speedup, which

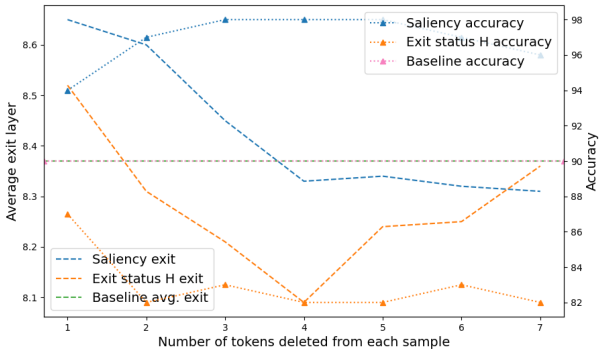
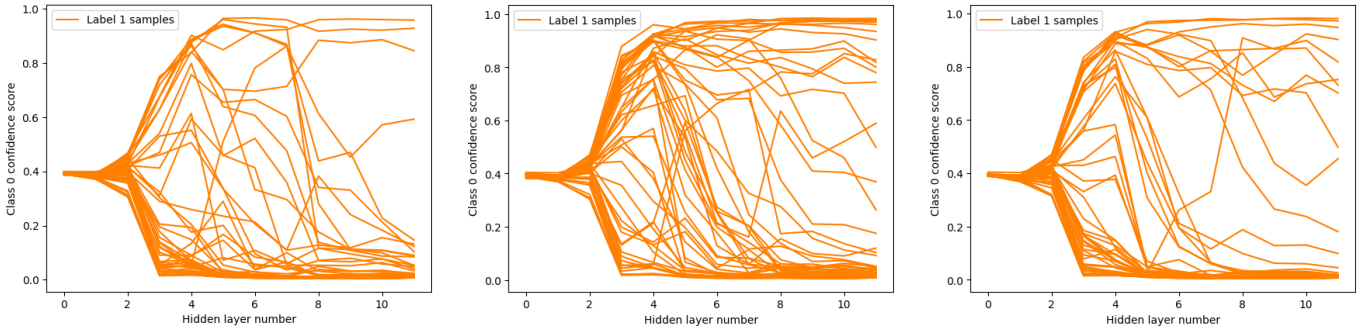


Fig. 3: Accuracy and average exit layer corresponding to SST-2 test data where various numbers of tokens with either the lowest saliency or smallest exit status H were deleted from the inputs, compared to baseline values associated with the original dataset (no deleted words). Exits were determined using patience-based criteria ($p = 5$). It is evident that saliency is more relevant to accuracy while exit status H is more relevant to exit layer, but both heuristics effect both metrics.

are understandable results considering the objectives (either misclassification or slowdown) that inspired these metrics' definitions. Still, the two metrics' similar effects provide additional support of the idea posed in Sec. IV-B1 that in the text domain, the misclassification and slowdown attack objectives are entangled. In Fig. 4, we plot the layerwise class confidence scores associated with class 0 for the 100 SST-2 samples that had class label 1 and had three words from the input text deleted according to either saliency or exit status H. For both the deletion of the least salient and the highest exit status H words, we see in Fig. 4 and that confidence scores are being pushed away from .5. As we explained in Sec. IV-B1, this is the value near which confidence scores would be representing the model's confusion in deciding on a classification. However, saliency favors samples being pushed towards the correct class label (for class 1 samples, this means class 0 confidence scores are pushed towards 0), while exit status H pushes samples towards either a correct or incorrect class, solely favoring a sample's minimized confusion. This means that using exit status H offers greater potential for creating more confident samples because it is not burdened by additional objectives. Additionally, it appears to inherently favor earlier high-confidence than saliency does, enabling a quicker establishment of the prediction consistency required for exiting with patience-based criteria. Many Fig. 9b confidence scores do not settle until around layers 5-8 with the use of saliency, compared to around 4-7 from use of exit status H in Fig. 4c. In Appendix C we include results associated with the deletion of the most salient and lowest exit status H words.



(a) Baseline (no words deleted). (b) Least salient words deleted. (c) Words with highest exit status H deleted.

Fig. 4: The confidence scores associated with class 0 for 100 SST-2 samples from class 1 (one line per sample) that had three words deleted, which were either those with the lowest saliency (b) or the highest exit status H (c), which we compare to the baseline confidence scores associated with the original data (a). The trajectories of these confidence scores demonstrate the fact that deletions corresponding either to saliency or exit status H will push confidence scores towards or away from a point of confusion (.5 confidence), but saliency also considers label correctness while exit status H favors having an early effect.

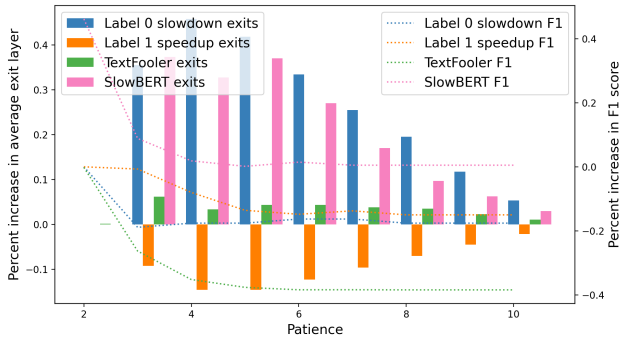


Fig. 5: Comparing increases in F1 score and average exit layer for SST-2 samples resulting from either modifications made by a TextFooler attack, SlowBERT, or modifications determined by an exit classifier to either slow down samples assigned exit label 0 or speed up samples assigned exit label 1. The classifier-determined slowdown modifications and the SlowBERT attack both cause more slowdown and less of a drop in F1 score than TextFooler, with SlowBERT causing the smallest F1 drop as it either maintains or increases baseline F1 scores. The classifier-determined speedup causes a decrease in exit layer and drop in F1 score only slightly less than that caused by classifier-determined slowdown. Figs. 10 and 11 in Appendix D display exit and F1 results separately.

C. Using Machine Learning to Study the Distinction between Slow and Fast Inputs

1) Defining a Novel Exit-Classification Task: In this section, we explore the idea of using a ML model to understand the slowdown effect for us. We show how the model's understanding can be exploited to generate a slowdown attack that can target a separate, fine-tuned, BERT-based MEM. Specifically, we train a model, which we refer to as an *exit classifier*, on the novel concept of an exit prediction task, which is a binary classification task where a model is provided with input text that it needs to classify as either an earlier-exiting sample (label 0) or later-exiting sample (label 1). For a specific downstream task, we craft a dataset wherein the input values are BERT embeddings (the penultimate hidden layer state taken from a pre-trained BERT base model that has not undergone any fine-tuning) of the text input from the original task. For the output values, we assign label 0 to samples with exit layers 1 through M and label 1 to samples with exit layers $M+N$ through K. Exit layers are determined by the target fine-tuned model and using the confidence-based exiting criterion using some threshold τ. K corresponds to the total number of layers in the target fine-tuned model, and M and N are tunable hyperparameters. The choice of M and N should enable each class to have sufficient data for the exit classifier to learn from. It is beneficial to maximize N, hence maximizing the gap in exits between the two classes, because the task should be easier to learn if the samples are more definitive examples of early-exiting and late-exiting texts.

2) Training an Exit Classifier for SST-2: For SST-2, to craft the exit prediction dataset, we used a threshold of τ = .9, assigned the original input samples' BERT embeddings associated with exit layers 1-4 to label 0, which equated to 37% of the original SST-2 training set and 29% of the test set, and embeddings associated with layers 8-12 were assigned to label 1, which was 5% of the original training set and 7% of the test set. We discarded all samples associated with any other exit layer. These choices of τ, M, and N, were deemed optimal based on a hyperparameter search. Essentially, assigning layer 1-4 to label 0 and 8-12 to label 1 appeared to be an option that resulted in a decent amount of samples for each class, but it also offered a wide gap between the two classes' exit layers, which should make the distinction between the classes more obvious and easier to learn. We used the resulting dataset to train a simple multi-layer perceptron

with one hidden layer followed by one dropout layer. After 20 epochs of training, the exit classifier converged at **.8198** train AUC (area under the ROC curve) and **.8344** test AUC. It is important to note that we evaluated SlowBERT-attacked SST-2 samples on the learned SST-2 exit classifier, expecting the model to classify them with exit label 1 to indicate that these samples are relatively late-exiting, but the performance was poor (resulted in **0.4346** AUC). This implies that what the exit classifier understands as distinguishing features between early-exiting and late-exiting samples are different from what is exploited by SlowBERT to cause slowdown. However, in the next subsection (IV-C3), we will use the SST-2 exit classifier not only to successfully cause slowdown, but also to cause speedup, therefore justifying the fact that the model does actually understand something that can be attributed to the relative speed at which texts can exit a MEM.

3) Crafting a Slowdown Attack Using an Exit Classifier: If an adversary can learn an exit classifier, then they can use it to craft a slowdown attack, though they would first need to craft an exit-classification dataset as described previously, which requires them to have knowledge of the exit layers that the target inputs take on the target model. However, for fine-tuned pre-trained BERT-based models, the adversary can likely use a pre-trained BERT model to estimate a fine-tuned model's exits. In Fig. 5, we plot results associated with a simple slowdown attack that was generated using the SST-2 exit classifier. We simply used the TextFooler attack generation process, but instead of aiming to misclassify original SST-2 samples, the goal of the attack was to misclassify label 0 (earlier-exiting) samples from the SST-2 exit prediction dataset as label 1 (later-exiting) samples. This means that the modifications made to the label 0 samples are theoretically turning earlier-exiting samples into late-exiting samples. Note that these modifications solely include synonym replacements. To be clear, the goal of this attack is not necessarily to cause misclassification for the exit-classification task. Rather, the misclassification objective is being used to guide a loss computation that will correspond with our true aim of modifying texts in such a way that early-exiting samples become later-exiting samples. In order to support the claim that the exit classifier is truly learning the distinction between early and late exiting samples, we also executed a sort of 'attack' (though not technically an attack because the intention is not to cause adverse effects) with the opposite objective to achieve the opposite result. In other words, we launched a TextFooler attack with the aim to misclassify label 1 (later-exiting) samples as label 0 (earlier-exiting) samples, which led to modifications that caused inference speedup. Examples from both the slowdown and speedup 'attacks' are included in in Table II. Fig. 5 compares both of these attack results to the traditional TextFooler attack that aims to cause misclassification and to SlowBERT, which aims to cause slowdown. However, our intention here is not necessarily to present an attack method that leads to a maximal slowdown effect, but rather to demonstrate the fact that an exit classifier has the potential to acquire its own understanding of what kinds of features contribute to some texts necessitating more inference than others.

V. Increasing Robustness against Slowdown through Expedited Adversarial Training

In this section, we show that a modified version of adversarial training, which we refer to as *expedited adversarial training*, can both mitigate inference slowdown on attacked samples and cause inference speedup on benign examples. [6] found that adversarial training was unsuccessful in mitigating slowdown attacks in the image domain, but we will explain how some of the behaviors associated with text slowdown that we discussed in Sec. IV-B1 can help explain why expedited adversarial training is effective in the text domain. Results in Sec. V-B show that this method, which is attack-agnostic, can mitigate slowdown effectively in six diverse attack settings.

A. Our Proposed Expedited Adversarial Training

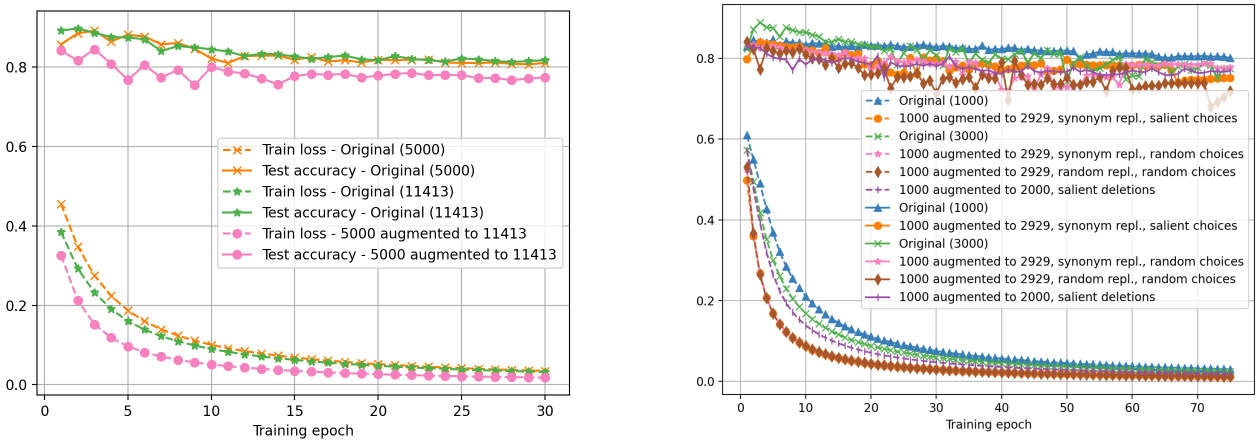
Our expedited adversarial training process involves fine-tuning a BERT base model with a training dataset that has been augmented with synthetic samples. We refer to the resulting model as a robust model. The synthetic samples are created by taking original data samples, choosing words in their input texts, and swapping those words with new words. In the next section, V-B, we explain the specific ways in which we choose words to replace and their replacements, but we will later demonstrate the fact that the specific means of choosing and replacing words is flexible and inconsequential. Note that in our experiments, we consistently choose to replace 20% of words because we have found that traditional text-based adversarial attacks often modify around 20% of tokens in a text in order to effectively attack it. These synthetic samples are not necessarily attacked samples because we do not ensure that the modifications we are making to inputs are leading to slower inference, which is why this process is not quite a traditional adversarial training.

B. Inference Speedup and Attack Mitigation Results

In Fig. 6a and Fig. 7a, across 30 total training epochs, we compare the test accuracy, training loss, and average exit layer associated with three different models fine-tuned on SST-2 data. The first is a baseline model that was trained on 5,000 original training samples. The second is a robust model that trained on a 11,413-sample dataset consisting of 5,000 original samples that each contributed an additional 1-2 synthetic samples through the replacement of their most salient words (using equation 1) with randomly-selected synonyms. The third is a baseline model that was trained on 11,413 original training samples. In Fig. 6b and Fig. 7b, across 75 training epochs, we compare the test accuracy, training loss, and average exit layer associated with six different models fine-tuned on MRPC data. We asses two baseline models, one trained on 1,000 original training samples and one trained on 3,000, and four robust models trained on datasets that include 1,000 original samples and 1-2 additional synthetic samples that were generated from the original 1,000 using different strategies. The first model replaced the most salient words with random

TABLE II: Examples of benign SST-2 input sentences paired with a version that was modified using an exit classifier in order to either cause slowdown or speedup.

Slowdown Sample 1	
Benign	allows **us** to hope that nolan is **poised** to embark a major career as a commercial yet inventive filmmaker.
Modified	allows **ourselves** to hope that nolan is **loaning** to embark a major career as a commercial yet inventive filmmaker.
Slowdown Sample 2	
Benign	you **do**n't have to know about **music** to appreciate the **film**'s easygoing blend of comedy and romance.
Modified	you **make**n't owns to know about **musica** to appreciate the **stills**'s easygoing blend of comedy and romance.
Speedup Sample 1	
Benign	the **emotions** are raw and **will** strike a nerve with anyone who's ever had family trauma.
Modified	the **sentiments** are raw and **dedication** strike a nerve with anyone who's ever made family trauma.
Speedup Sample 2	
Benign	we **know** the plot's a little **crazy**, but it **held** my interest from **start** to finish.
Modified	we **realizing** the plot's a little **wack**, but it **possessed** my interest from **opened** to finish.

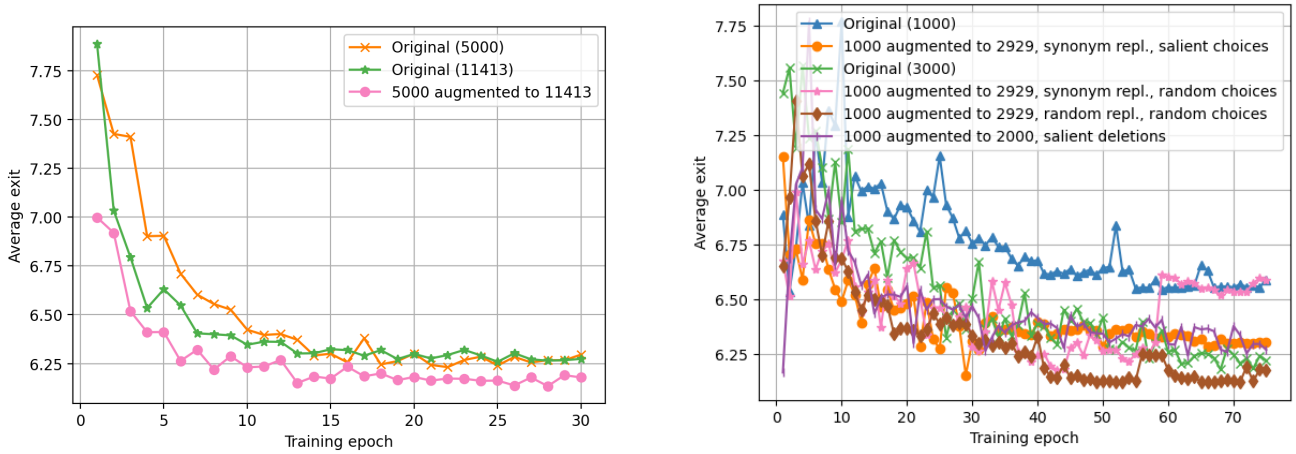


(a) SST-2 (b) MRPC

Fig. 6: Comparison of train loss and test accuracy associated with various baseline models (trained only using original data) and robust models (trained with a mix of original data and synthetic data) throughout training. All of the robust models exhibit slightly lower accuracy and loss than the baseline for both SST-2 and MRPC.

synonyms, the second model replaced randomly-chosen words with random synonyms, the third replaced randomly-chosen words with random words (thus compromising the sentence's meaning), and the fourth deleted the most salient words from the sentence. We can draw three main conclusions from these results. First, training with synthetic samples leads to inference speedup (decreased average exit layer) for benign samples without causing a notable drop in accuracy. Second, this effect is comparable across all sample-creation strategies, including replacing either salient words or randomly-chosen words and using either synonyms and random words as replacements. Lastly, the beneficial decreased average exit reaped through the use of these augmented datasets cannot simply be attributed to an increased training dataset size.

In Table III, we show results associated with SST-2 and MRPC that compare a baseline model and two robust models in terms of the F1 score and average exit layer computed on each task's benign test dataset and on three different 'attacked' datasets per task. Results associated with MNLI are in Table VIII in Appendix E. These attacked datasets consist of samples that have modified original samples in some way that either intentionally or unintentionally induces slowdown. For this table, the baseline models are models that were trained on the original SST-2 and MRPC training datasets. For both tasks, we report results associated with a robust model whose synthetic training samples made random replacements for randomly-selected words and a robust model whose synthetic samples replaced the most salient words with random synonyms. For all attacks and for both datasets, the average exit layer is drastically smaller for both robust models than for the baseline mode. In fact, all average exit layers associated with the robust models only barely surpass 6.00. Since we set p to 5 in these experiments, the minimum possible exit layer that any sample can take is 6, which means the robust



(a) SST-2 (b) MRPC

Fig. 7: Comparison of the average exit layer taken by all test samples when being evaluated on various baseline models (trained only using original data) and robust models (trained with a mix of original data and synthetic data) throughout training. For both SST-2 and MRPC, the robust models generally experience earlier exiting than the baseline.

TABLE III: Comparing the F1 score and average exit layer when evaluating a variety of attacked SST-2 and MRPC datasets on a baseline model trained on the original training datasets and two robust models that were trained with expedited adversarial training (trained on augmented datasets). The attacked datasets were produced using either the SlowBERT attack, the exit-classification based slowdown attack introduced in Sec. IV-C3, pre-existing text misclassification attacks (TextFooler [19] or PWWS [17]), or replacing a random 20% of words with randomly-selected synonyms. We consistently use patience-based exiting criteria with $p = 5$. The results show that, compared to the baseline model, both robust models for both datasets exhibit decreased F1 score on the benign data, either increased or decreased F1 score across the various 'attacks', but consistently decreased exit layer that nearly reaches the minimal possible value of 6 ($p = 5$ does not enable exiting earlier than layer 6).

SST-2								
Model	Benign		SlowBERT-attacked		Exit-classification slowdown		TextFooler-attacked	
	F1 score	Avg. exit	F1 score	Avg. exit	F1 score	Avg. exit	F1 score	Avg. exit
Baseline	0.87	8.23	0.90	11.49	0.81	8.60	0.87	8.79
Robust (salient words, synonym replacement)	0.79	6.16	0.96	6.13	0.91	6.14	0.68	6.20
Robust (random words, random replacement)	0.80	6.22	0.74	6.42	0.88	6.16	0.61	6.25
MRPC								
Model	Benign		SlowBERT-attacked		Random replacement		PWWS-attacked	
	F1 score	Avg. exit	F1 score	Avg. exit	F1 score	Avg. exit	F1 score	Avg. exit
Baseline	0.86	7.24	0.76	10.39	0.75	8.20	0.65	8.03
Robust (salient words, synonym replacement)	0.77	6.33	0.79	6.21	0.75	6.16	0.82	6.26
Robust (random words, random replacement)	0.75	6.24	0.75	6.34	0.75	6.20	0.78	6.20

models are nearly allowing for the maximal efficiency benefit that the MEM is capable of providing.

C. Linking the Expedited Adversarial Training Success to the Overlap between Misclassification and Slowdown Objectives

The results from Sec. V-B show that, regardless of the specific augmentation strategy chosen, augmenting a training dataset with samples that have had words swapped in the input text will lead to quicker inference time on benign data without notably compromising accuracy and will mitigate the slowdown effect of attacked data. The similarity of results from the various augmentation strategies is crucial because traditional adversarial training is characteristically expensive, but we've shown that the quick process of replacing random words with new random words can be sufficient for achieving robustness against slowdown in the text domain.

We suppose that the efficacy is largely due to a unique sort of regularization effect. The synthetic training samples can be considered noise, and it is known that training a model with more noisy samples biases the model towards ignoring extra noise terms in its hypothesis function. In this scenario,

when we introduce the 'noisy' synthetic samples to the training data, which we can assume is slightly less-accurate, harder-to-learn-from data than the original samples, it makes sense that the 'robust' model trained on these synthetic samples will yield slightly less-accurate predictions on benign samples than the baseline model. Yet, the incurred regularization effect will mean that the robust model is more confident in its predictions, more quickly than the baseline model. This aligns with the observation we made in Paragraph IV-B2b that word deletion guided by the exit status H leads to less potential for achieving maximal accuracy and more potential for causing speedup because it favors a model's confidence in assigning predictions to a sample over its correctness in the assignments. Not only is the regularization effect causing increased model confidence when running inference on benign samples, but it seems logical that it would also offer less opportunity for an attacker to craft an adversarial sample that confuses the model. Since we addressed in Sec. IV-B1 the fact that the slowdown effect comes from model confusion as opposed to a model's prediction oscillating between confidently leaning towards different classes, the lessened opportunity for an attacker to confuse a model would understandably lead to the mitigation of the slowdown effect of adversarial samples. Lastly, it's worth noting that the robust models did slightly mitigate the accuracy degradation that the attacks in Table III caused, but not as effectively as the slowdown was mitigated. This supports the idea that text slowdown is both easier to cause, therefore is sometimes caused unintentionally, and easier to defend against, therefore benefiting from a simplified version of adversarial training that is not sophisticated enough to combat misclassification.

As a final possible explanation, it is plausible that the expedited adversarial training process offers a model more experience with processing and deriving predictions for more confusing texts. The synthetic samples are created in a relatively haphazard way and with no assurance that the resulting sentences are logical. Even when synonyms are used as replacements for words, simply choosing a replacement word with a similar meaning as the original word does not mean that the new word will be coherent in the context it is placed in. This means that a model, after training on these samples, may learn to be more adept in processing confusing samples, thus explaining the earlier exiting that we see with the robust models. Recent works explain that BERT processes different attributes of a text at different layers [55], [56]. Therefore, we hypothesize that expedited adversarial training leads to a model whose layerwise considerations are more favorable to deriving confident predictions for confusing samples early in the network. For instance, perhaps a model will see these synthetic samples with confusing syntax and grammar and therefore attribute fewer layers to processing these attributes, consequently shifting the task-specific processing that usually happens later up to earlier layers in the network. With task-specific processing occurring earlier in the network, a model can come to a decision about the task earlier in the network and therefore allow inputs to exit earlier.

VI. Discussion

We have presented an investigation of the ways in which a slowdown effect can be inflicted upon and observed within BERT-based MEMs and we used our findings to justify a surprisingly simple, yet highly effective means of combating the slowdown. Our results can serve as a basis for further study towards better understanding the phenomena that are confusing to language models. A fundamental distinction between slowdown attacks and traditional adversarial attacks (i.e. misclassification attacks) that is worth emphasis is as follows. Typical adversarial attacks aim to fool a model into some incorrect output, while slowdown attacks do not care about the specific output and are instead concerned with a model's quickness and certainty in its derivation of the output. This distinction is likely at the root of all of the unique patterns that we see with slowdown attacks. Essentially, our empirical findings suggest that slowdown can be associated with any vague notion of confusion or uncertainty, which is easy to impose in such a constrained, nuanced domain (text), yet also means that the simple act of forcing a model to be more confident can be highly effective in mitigating slowdown. On one hand, this contrasts traditional text-based attacks whose goals are harder to impose and thus harder to mitigate. This also contrasts patterns of slowdown that occur in the vision domain because the slowdown effect has shown to require more specific intention to impose, which could also explain the failure of adversarial training as a mitigation strategy for image-based slowdown. One area we find particularly worthy of further exploration is the way that different sources of slowdown are present in layerwise confidence score patterns. Recent works suggest that BERT considers different qualities of text input across different layers of the model in a way that doesn't align with the understanding that image models consider simpler image features in earlier layers and more complex features in later layers [55], [56]. Linking the knowledge of what BERT understands at each layer with knowledge of what causes slowdown and how is evidenced across model layers can inform new slowdown attack generation approaches and more slowdown-robust methods. As mentioned previously, we expect there to be some interplay between a model's vulnerability to slowdown and the layers at which the model processes certain attributes of a text, and our proposed expedited adversarial training may be taking advantage of or adapting this interplay. Still, it will be important for future work to identify new attack methods that are resilient to any form of adversarial training. Ultimately, as we continue to broaden our understanding of what causes inference slowdown in the language domain, we can continue to improve the design of models in a way that favors efficiency, which is vital to a field with models and datasets that experiences relentless expansion.

Acknowledgments

We thank Naomi Feldman, Colin Philips and the anonymous reviewers for their constructive feedback. This research was supported by a grant from the Department of Defense.

REFERENCES

[1] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[2] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," 2021.

[3] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "Deebert: Dynamic early exiting for accelerating bert inference," 2020.

[4] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," 2020.

[5] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, "Fastbert: a self-distilling bert with adaptive inference time," 2020.

[6] S. Hong, Y. Kaya, I.-V. Modoranu, and T. Dumitraş, "A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference," in *International Conference on Learning Representations*, 2021.

[7] S. Zhang, X. Pan, M. Zhang, and M. Yang, "SlowBERT: Slow-down attacks on input-adaptive multi-exit BERT," in *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 9992–10 007.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[9] J. Y. Yoo and Y. Qi, "Towards improving adversarial training of nlp models," 2021.

[10] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," 2018.

[11] S. Garg and G. Ramakrishnan, "BAE: BERT-based adversarial examples for text classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.

[12] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," 2020.

[13] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan, "Contextualized perturbation for textual adversarial attack," 2021.

[14] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," 2018.

[15] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," 2019.

[16] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Word-level textual adversarial attacking as combinatorial optimization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 6066–6080.

[17] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1085–1097.

[18] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating adversarial text against real-world applications," in *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019.

[19] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," 2020.

[20] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," 2020.

[21] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," 2018.

[22] Y. Kaya, S. Hong, and T. Dumitraş, "Shallow-Deep Networks: Understanding and mitigating network overthinking," in *Proceedings of the 2019 International Conference on Machine Learning (ICML)*, Long Beach, CA, Jun 2019.

[23] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Hk2aImxAb

[24] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 274–19 286.

[25] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[26] OpenAI, "Gpt-4 technical report," 2023.

[27] S. Chen, Z. Song, M. Haque, C. Liu, and W. Yang, "Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models," 2022.

[28] S. Chen, H. Chen, M. Haque, C. Liu, and W. Yang, "The dark side of dynamic routing neural networks: Towards efficiency backdoor injection," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2023, pp. 24 585–24 594.

[29] J. Pan, Q. Zheng, Z. Fan, H. Rahmani, Q. Ke, and J. Liu, "Gradauto: Energy-oriented attack on dynamic neural networks," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 637–653.

[30] H. Liu, Y. Wu, Z. Yu, Y. Vorobeychik, and N. Zhang, "Slowlidar: Increasing the latency of lidar-based detection using adversarial examples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 5146–5155.

[31] A. Shapira, A. Zolfi, L. Demetrio, B. Biggio, and A. Shabtai, "Phantom sponges: Exploiting non-maximum suppression to attack deep object detectors," 2022.

[32] J. Pan, L. G. Foo, Q. Zheng, Z. Fan, H. Rahmani, Q. Ke, and J. Liu, "Gradmdm: Adversarial attack on dynamic networks," 2023.

[33] M. Ayyat, S. K. Nukavarapu, and T. Nadeem, "Dynamic deep neural network adversarial attacks for edge-based iot devices," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 61–67.

[34] I. Shumailov, Y. Zhao, D. Bates, N. Papernot, R. Mullins, and R. Anderson, "Sponge examples: Energy-latency attacks on neural networks," 2021.

[35] N. Boucher, I. Shumailov, R. Anderson, and N. Papernot, "Bad characters: Imperceptible nlp attacks," 2021.

[36] X. Wang, H. Jin, Y. Yang, and K. He, "Natural language adversarial defense through synonym encoding," 2021.

[37] L. Li and X. Qiu, "Token-aware virtual adversarial training in natural language understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 8410–8418.

[38] E. Altinisik, H. Sajjad, H. T. Sencar, S. Messaoud, and S. Chawla, "Impact of adversarial training on robustness and generalizability of language models," *arXiv preprint arXiv:2211.05523*, 2022.

[39] X. Liu, H. Cheng, P. He, W. Chen, Y. Wang, H. Poon, and J. Gao, "Adversarial training for large neural language models," *arXiv preprint arXiv:2004.08994*, 2020.

[40] D. N. Rim, D. Heo, and H. Choi, "Adversarial training with contrastive learning in nlp," *arXiv preprint arXiv:2109.09075*, 2021.

[41] M. Haque, Y. Yadlapalli, W. Yang, and C. Liu, "EREBA," in *Proceedings of the 44th International Conference on Software Engineering*. ACM, may 2022.

[42] S. Geng, P. Gao, Z. Fu, and Y. Zhang, "Romebert: Robust training of multi-exit bert," 2021.

[43] A. Mehra, S. Seto, N. Jaitly, and B.-J. Theobald, "Understanding the robustness of multi-exit models under common corruptions," 2022.

[44] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[45] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," 2019.

[46] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642.

[47] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[48] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers). Association for Computational Linguistics, 2018, pp. 1112–1122.
[49] G. A. Miller, "WordNet: A lexical database for English," in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. [Online]. Available: https://aclanthology.org/H94-1111
[50] J. A. Michaelov, M. D. Bardolph, S. Coulson, and B. K. Bergen, "Different kinds of cognitive plausibility: why are transformers better than rnns at predicting n400 amplitude?" *arXiv preprint arXiv:2107.09648*, 2021.
[51] K. Krishna, Y. Song, M. Karpinska, J. Wieting, and M. Iyyer, "Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense," *arXiv preprint arXiv:2303.13408*, 2023.
[52] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.
[53] P. Damodaran, "Parrot: Paraphrase generation for nlu." 2021.
[54] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 119–126.
[55] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?" in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3651–3657.
[56] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," 2020.

APPENDIX

A. Comparing the Effectiveness of SlowBERT Modification Types While Varying the Modification Allowance

As an additional experiment to help us identify the most effective SlowBERT modification types, we executed the SlowBERT attack with a constraint on the maximum number of words that could be altered. A summary of the results of this experiment, using both SST-2 and MRPC data, is visually depicted in Table IV. These results align with the conclusions we made in Sec. IV-B2. We see that, across all numbers of allowed modifications, character-level modifications have higher potential to cause slowdown than Sub-W (a word-level modification) and Insert is the most effective character-level modification.

B. Comparing the Most Certain Confidence Scores to the Least Certain Confidence Scores

Here, we include two additional plots to help illustrate part of the reasoning we used in Sec. IV-B1 to explain why misclassification attacks lead to an inadvertent slowdown effect due to the fact that the achieving the misclassification objective often happens through some form of model confusion. With Fig. 8a we see that, especially with label 0 samples, even the eventual misclassification (low confidence score for class 0 at the output layer) involves correct classification in earlier layers (high confidence scores for class 0 around layers 3-7). This pattern of switching from favoring one class to favoring the other impedes the model's ability to establish consistency in its prediction that is required for a sample to exit early. With Fig. 8b, we see that, especially compared to the more-certain samples, the less-certain samples both have confidence scores around .5 at the output layer and at earlier layers. This is demonstrating the fact that samples that are confusing to the model (in other words, the model cannot confidently assign them to either class) at the final output layer will usually also be confusing to the model in preceding layers. This again means that the model is unable to establish the consistent confidence in a single classification that would enable early exiting.

C. Results Associated with Heuristic-Guided Token Deletion

In this section, we include results to accompany those in Fig. 4 of Sec. IV-B2b. In Fig. 4 we can observe that, compared to the baseline, deleting words with either the smallest saliency or highest exit status H causes samples' confidence scores to be pushed towards higher confidence. Saliency evidently favors higher confidence towards an incorrect classification while exit status H is solely maximizing confidence in one classification's direction with no care about the correctness of the classification. Now, in Fig. 9, we present results associated with deletion of the most salient or words or those with the smallest exit status H and see that both scenarios are pushing confidence scores towards the middle of the plot (.5 confidence) thereby avoiding confident assignment to either label.

D. Additional Classifier-Based Slowdown Attack Results

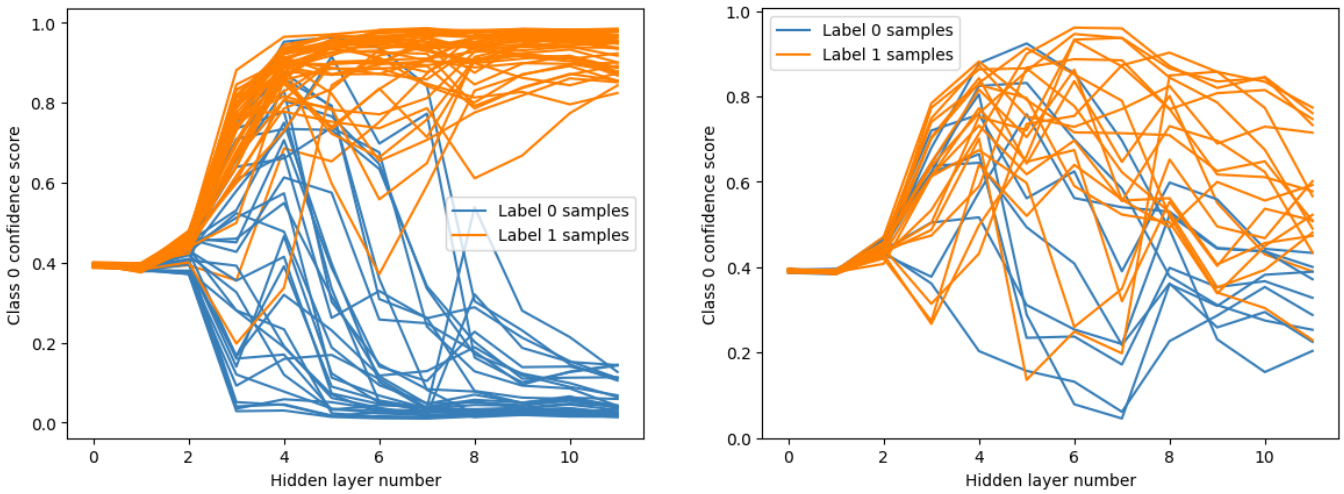
In the main text, when comparing our novel, exit-classification-based slowdown attack to TextFooler and SlowBERT baselines, we combine the average exit results and F1 score results into a single plot. Here, we present the exit results and F1 results in separate plots in case these plots provide more clarity.

E. Multi-Genre Natural Language Inference Results

Here, we provide a variety of results associated with the Multi-Genre Natural Language Inference dataset [48] from the General Language Understanding Evaluation (GLUE) benchmark. MNLI is a task where two sentences, a premise sentence and a hypothesis sentence, are assigned an 'entailment' label if the premise entails the hypothesis, a 'contradiction' label if the hypothesis contradicts the premise, or a 'neutral' label if neither aforementioned case is true.

TABLE IV: The percentage of attack executions in which each SlowBERT modification type was chosen (due to offering the most slowdown) by SlowBERT attacks on the SST-2 and MRPC test datasets. We consider versions of the attack that limit the number of modifications that can be made to the input text in addition to the default version of the attack, which continues making modifications until the resulting attacked sample exits at the last possible exit layer or there are no words left to modify. All experiments used patience-based exiting criteria with $p = 6$. Generally, character-level modifications cause more slowdown than Sub-W (a word-level modification) and the increase in exit layer increases as more modifications are allowed.

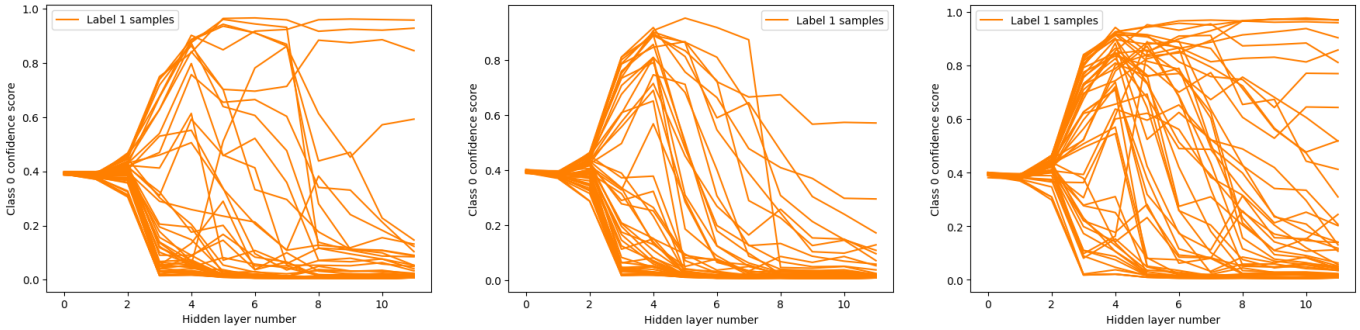
Task	Max number of modifications allowed	Usage percentage by modification type (%)						Avg. exit
		Original	Insert	Delete	Swap	Sub-C	Sub-W	
SST-2	1	3.19	38.25	20.98	13.68	19.12	4.78	9.950
	2	5.66	37.52	18.55	13.14	18.97	6.16	10.727
	3	8.34	36.00	18.03	12.72	18.30	6.59	11.168
	4	9.66	35.00	17.29	12.58	18.66	6.79	11.410
	Unlimited	16.95	33.01	15.31	10.69	16.05	7.98	11.932
MRPC	1	10.49	33.76	14.76	20.58	12.64	7.73	9.06
	2	13.93	31.79	14.16	18.99	12.56	8.56	9.721
	3	16.66	30.11	13.68	18.24	12.69	8.61	10.204
	4	19.25	29.19	13.02	17.31	12.55	8.67	10.554
	Unlimited	35.75	21.93	10.04	11.29	11.13	9.87	11.589



(a) Samples with the most certain confidence scores (class 0 confidence $>$ .8 or $<$ .2).

(b) Samples with the least certain confidence scores (class 0 confidence $>$ .2 and $<$ .8).

Fig. 8: The confidence scores associated with class 0 for 100 SST-2 samples (one line per sample, color-coded according to correct class label), taken from each layer. We compare the most certain (a) and uncertain (b) values resulting from a successful TextFooler attack in order to demonstrate the patterns of model confusion that we observe as a side effect of misclassification attacks and that can explain the inadvertent slowdown that accompanies misclassification attacks.



(a) Baseline (no words deleted). (b) Most salient words deleted. (c) Words with lowest exit status H deleted.

Fig. 9: The confidence scores associated with class 0 for 100 SST-2 samples from class 1 (one line per sample) that had 3 words deleted, which were either those with the highest saliency (a), lowest saliency (b), lowest exit status H (c), or highest exit status H (d). The trajectories of these confidence scores demonstrate the fact that deletions corresponding either to saliency or exit status H will push confidence scores towards or away from a point of confusion (.5 confidence), but saliency also considers label correctness while exit status H favors having an early effect.

TABLE V: Accuracy (Acc.) and average exit layer (Ext.) corresponding to versions of the SST-2 test dataset where various numbers of tokens were deleted according to the following heuristics: either the words with the lowest saliency or smallest exit status H were deleted from the text inputs. We compare these values to the baseline values associated with the original dataset (no tokens deleted). Exits were determined using patience-base criteria with p = 5. It is evident that saliency is more relevant to accuracy while exit status H is more relevant to exit layer, but both heuristics effect both metrics. Note that without any deletions, this dataset achieves an average exit of 8.37 and 90% accuracy.

Heuristic	Metric	# of tokens deleted						
		1	2	3	4	5	6	7
Saliency	Ext.	8.65	8.6	8.45	8.33	8.34	8.32	8.31
	Acc.	94	97	98	98	98	97	96
Exit status H	Ext.	8.52	8.31	8.21	8.09	8.24	8.25	8.36
	Acc.	87	82	83	82	82	83	82

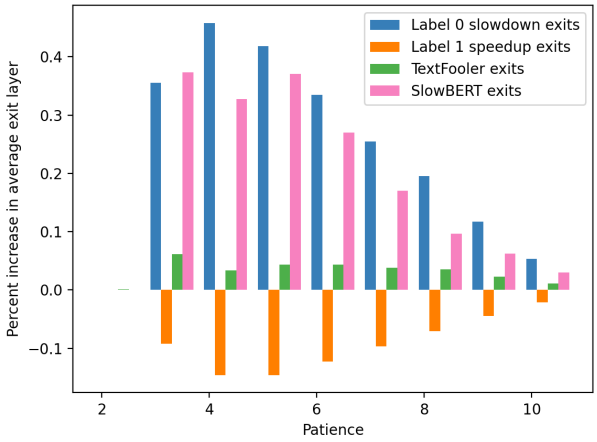


Fig. 10: Comparison of percent increase in average exit layer for SST-2 samples resulting from either modifications made by a TextFooler attack, SlowBERT, or modifications determined by an exit classifier to either slow down samples assigned exit label 0 or speed up samples assigned exit label 1. The classifier-determined slowdown modifications and the SlowBERT attack both cause more slowdown than TextFooler while the classifier-determined speedup causes a decrease in exit layer.

TABLE VI: In order to communicate crucial differences between saliency and exit status H, we compare confidence scores associated with class 0 at each layer for samples where the three words with either the lowest exit status H, highest exit status H, lowest saliency, or highest saliency were deleted. We report the percentage of total samples that are 'near .5' (confidence $> .2$ or $< .8$), the percentage of samples that are 'far from .5' (confidence $< .2$ or $> .8$), and the percentage of samples whose confidence score is indicating a correct classification.

Deleted	Metric	Layer #								
		3	4	5	6	7	8	9	10	11
Baseline (no deletion)	% near .5	34.62	15.38	23.08	17.31	9.62	9.62	7.69	5.77	1.92
	% far from .5	65.38	84.62	76.92	82.69	90.38	90.38	92.31	94.23	98.08
	% correct	80.77	73.08	80.77	80.77	84.62	92.31	94.23	92.31	92.31
Lowest exit status H	% near .5	57.69	23.08	26.92	21.15	25.0	26.92	25.0	21.15	23.08
	% far from .5	42.31	76.92	73.08	78.85	75.0	73.08	75.0	78.85	76.92
	% correct	36.54	34.62	44.23	55.77	59.62	69.23	73.08	75.0	76.92
Highest exit status H	% near .5	36.54	15.38	13.46	17.31	7.69	9.62	11.54	9.62	13.46
	% far from .5	63.46	84.62	86.54	82.69	92.31	90.38	88.46	90.38	86.54
	% correct	65.38	65.38	73.08	80.77	82.69	82.69	84.62	84.62	86.54
Lowest saliency	% near .5	57.69	25.0	30.77	30.77	19.23	13.46	13.46	11.54	11.54
	% far from .5	42.31	75.0	69.23	69.23	80.77	86.54	86.54	88.46	88.46
	% correct	46.15	48.08	59.62	71.15	75.0	76.92	80.77	78.85	80.77
Highest saliency	% near .5	48.08	21.15	19.23	15.38	11.54	13.46	7.69	5.77	3.85
	% far from .5	51.92	78.85	80.77	84.62	88.46	86.54	92.31	94.23	96.15
	% correct	69.23	65.38	80.77	86.54	92.31	98.08	98.08	98.08	98.08

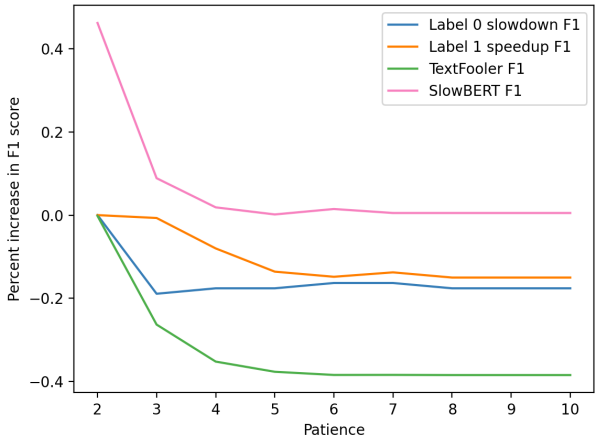


Fig. 11: Comparison of percent increase in F1 score for SST-2 samples resulting from either modifications made by a TextFooler attack or modifications determined by an exit classifier to either slow down samples assigned exit label 0 or speed up samples assigned exit label 1. The classifier-determined slowdown modifications and the SlowBERT attack both cause less of a drop in F1 score than TextFooler, with SlowBERT causing the smallest F1 drop as it either maintains or increases baseline F1 scores. The classifier-determined speedup causes only a slightly smaller drop in F1 score than the classifier-determined slowdown.

TABLE VII: Comparing the average exit layer taken by 100 MNLI test samples before and after the execution of three pre-existing misclassification attacks (TextFooler [19], probability weighted word saliency (PWWS) [17], and BERT-based Adversarial Examples (BAE) [11]) and the SlowBERT attack [7] in order to demonstrate the inadvertent slowdown effect that misclassification attacks cause. Each of the 100 samples were selected from a task's test dataset based whether the specific attack could be successfully generated from the sample input, therefore each attack uses a slightly different set of 100 samples. Patience-based exiting criteria was used, with $p = 5$.

Attack	Avg. exit pre-attack	Avg. exit post-attack	% change
TextFooler	7.13	8.16	14.45
PWWS	7.20	8.50	18.06
BAE	7.14	8.46	18.49
SlowBERT	7.40	12.00	62.16

TABLE VIII: Comparing the accuracy and average exit layer when evaluating the benign MNLI test dataset and a SlowBERT-attacked version of the MNLI test dataset on a baseline model trained on the original training dataset and a robust model that was trained with expedited adversarial training (trained on an augmented dataset that included synthetic samples where, in this case, randomly-selected words were replaced with randomly-selected synonyms). We consistently use patience-based exiting criteria with $p = 5$. The results show that, compared to the baseline model, the robust model exhibits decreased accuracy on the benign data, increased accuracy on the attacked data, and decreased average exit layers for both the benign and attacked datasets.

Model	Benign		SlowBERT-attacked	
	Accuracy	Average exit	Accuracy	Average exit
Baseline	0.89	7.40	0.62	12.00
Robust	0.71	7.06	0.60	8.14