Query-Efficient and Scalable Black-Box Adversarial Attacks on Discrete Sequential Data via Bayesian Optimization

Deokjae Lee [1] Seungyong Moon [1] Junhyeok Lee [1] Hyun Oh Song [1]

Abstract

We focus on the problem of adversarial attacks against models on discrete sequential data in the black-box setting where the attacker aims to craft adversarial examples with limited query access to the victim model. Existing black-box attacks, mostly based on greedy algorithms, find adversarial examples using pre-computed key positions to perturb, which severely limits the search space and might result in suboptimal solutions. To this end, we propose a query-efficient black-box attack using Bayesian optimization, which dynamically computes important positions using an automatic relevance determination (ARD) categorical kernel. We introduce block decomposition and history subsampling techniques to improve the scalability of Bayesian optimization when an input sequence becomes long. Moreover, we develop a post-optimization algorithm that finds adversarial examples with smaller perturbation size. Experiments on natural language and protein classification tasks demonstrate that our method consistently achieves higher attack success rate with significant reduction in query count and modification rate compared to the previous state-of-the-art methods.

1. Introduction

In recent years, deep neural networks on discrete sequential data have achieved remarkable success in various domains including natural language processing and protein structure prediction, with the advent of large-scale sequence models such as BERT and XLNet (Devlin et al., 2019; Yang et al., 2019). However, these networks have exhibited vulnerability against adversarial examples that are artificially crafted to raise network malfunction by adding perturbations imperceptible to humans (Papernot et al., 2016; Jin et al., 2020). Recent works have focused on developing adversarial attacks in the *black-box* setting, where the adversary can only observe the predicted class probabilities on inputs with a limited number of queries to the network (Alzantot et al., 2018; Ren et al., 2019). This is a more realistic scenario since, for many commercial systems (Google Cloud NLP, 2022; Amazon Comprehend, 2022), the adversary can only query input sequences and receive their prediction scores with restricted resources such as time and cost.

While a large body of works has proposed successful black-box attacks in the image domain with continuous attack spaces (Ilyas et al., 2018; Andriushchenko et al., 2020), developing a query-efficient black-box attack on discrete sequential data is quite challenging due to the discrete nature of their attack spaces. Some prior works employ evolutionary algorithms for the attack, but these methods require a large number of queries in practice (Alzantot et al., 2018; Zang et al., 2020). Most of the recent works are based on greedy algorithms which first rank the elements in an input sequence by their importance score and then greedily perturb the elements according to the pre-computed ranking for query efficiency (Ren et al., 2019; Jin et al., 2020; Maheshwary et al., 2021). However, these algorithms have an inherent limitation in that each location is modified at most once and the search space is severely restricted (Yoo et al., 2020).

To this end, we propose a *Blockwise Bayesian Attack* (BBA) framework, a query-efficient black-box attack based on *Bayesian Optimization*. We first introduce a categorical kernel with automatic relevance determination (ARD), suited for dynamically learning the importance score for each categorical variable in an input sequence based on the query history. To make our algorithm scalable to a high-dimensional search space, which occurs when an input sequence is long, we devise block decomposition and history subsampling techniques that successfully improve the query and computation efficiency without compromising the attack success rate. Moreover, we propose a post-optimization algorithm that reduces the perturbation size.

We validate the effectiveness of BBA in a variety of datasets

[1]Department of Computer Science and Engineering, Seoul National University, Seoul, Korea. Correspondence to: Hyun Oh Song <hyunoh@snu.ac.kr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

from different domains, including text classification, textual entailment, and protein classification. Our extensive experiments on various victim models, ranging from classical LSTM to more recent Transformer-based models (Hochreiter & Schmidhuber, 1997; Devlin et al., 2019), demonstrate state-of-the-art attack performance in comparison to the recent baseline methods. Notably, BBA achieves higher attack success rate with considerably less modification rate and fewer required queries on all experiments we consider.

2. Related Works

2.1. Black-Box Attacks on Discrete Sequential Data

Black-box adversarial attacks on discrete sequential data have been primarily studied in natural language processing (NLP) domain, where an input text is manipulated at word levels by substitution (Alzantot et al., 2018). A line of research exploits greedy algorithms for finding adversarial examples, which defines the word replacement order at the initial stage and greedily replaces each word under this order by its synonym chosen from a word substitution method (Ren et al., 2019; Jin et al., 2020; Maheshwary et al., 2021; Garg & Ramakrishnan, 2020; Li et al., 2020). Ren et al. (2019) determine the priority of words based on word saliency and construct the synonym sets using WordNet (Fellbaum, 1998). Jin et al. (2020) construct the word importance ranking by measuring the prediction change after deleting each word and utilize the word embedding space from Mrkšić et al. (2016) to identify the synonym sets. The follow-up work of Maheshwary et al. (2021) proposes a query-efficient word ranking algorithm that leverages attention mechanism and locality-sensitive hashing. Another research direction is to employ combinatorial optimizations for crafting adversarial examples (Alzantot et al., 2018; Zang et al., 2020). Alzantot et al. (2018) generate adversarial examples via genetic algorithms. Zang et al. (2020) propose a particle swarm optimization-based attack (PSO) with a word substitution method based on sememes using HowNet (Dong et al., 2010).

2.2. Bayesian Optimization

While Bayesian optimization has been proven to be remarkably successful for optimizing black-box functions, its application to high-dimensional spaces is known to be notoriously challenging due to its high *query complexity*. There has been a large body of research that improves the query efficiency of high-dimensional Bayesian optimization. One major approach is to reduce the effective dimensionality of the objective function using a sparsity-inducing prior for the scale parameters in the kernel (Oh et al., 2019; Eriksson & Jankowiak, 2021). Several methods address the problem by assuming an additive structure of the objective function and decomposing it into a sum of functions in lower-dimensional disjoint subspaces (Kandasamy et al., 2015; Wang et al., 2018b). Additionally, a line of works proposes methods that perform multiple evaluation queries in parallel, also referred to as batched Bayesian optimization, to further accelerate the optimization (Azimi et al., 2010; Wang et al., 2017).

Another challenge in Bayesian optimization with Gaussian processes (GPs) is its high *computational complexity* of fitting surrogate models on the evaluation history. A common approach to this problem is to use a subset of the history to train GP models (Seeger et al., 2003; Chalupka et al., 2012). Seeger et al. (2003) greedily select a training point from the history that maximizes the information gain. Chalupka et al. (2012) choose a subset of the history using Farthest Point Clustering heuristic (Gonzalez, 1985).

Many Bayesian optimization methods have focused on problem domains with continuous variables. Recently, Bayesian optimization on categorical variables has attained growing attention due to its broad potential applications to machine learning. Baptista & Poloczek (2018) use Bayesian linear regression as surrogate models for black-box functions over combinatorial structures. Oh et al. (2019) propose a Bayesian optimization method for combinatorial search spaces using GPs with a discrete diffusion kernel.

2.3. Adversarial Attacks via Bayesian Optimization

Several works have proposed query-efficient adversarial attacks using Bayesian optimization in image and graph domains, but its applicability to discrete sequential data has not yet been explored. Shukla et al. (2019); Ru et al. (2020) leverage Bayesian optimization to attack image classifiers in a low query regime. Shukla et al. (2019) introduce a noise upsampling technique to reduce the input dimensions of image spaces for the scalability of Bayesian optimization. A concurrent work of Ru et al. (2020) proposes a new upsampling method, whose resize factor is automatically determined by the Bayesian model selection technique, and adopts an additive GP as a surrogate model to further reduce the dimensionality. Recently, Wan et al. (2021) propose a query-efficient attack algorithm against graph classification models using Bayesian optimization with a sparse Bayesian linear regression surrogate. While these Bayesian optimization-based methods find adversarial examples with any perturbation size below a pre-defined threshold, we further consider minimizing the perturbation size, following the practice in the prior works in NLP (Ren et al., 2019; Jin et al., 2020; Zang et al., 2020; Maheshwary et al., 2021).

3. Preliminaries

3.1. Problem Formulation

To start, we introduce the definition of adversarial attacks on discrete sequential data. Suppose we are given a target classifier $f_\theta : \mathcal{X}^l \rightarrow \mathbb{R}^{|\mathcal{Y}|}$, which takes an input sequence of l elements $s = [w_0, \ldots, w_{l-1}] \in \mathcal{X}^l$ and outputs a logit vector used to predict its ground-truth label $y \in \mathcal{Y}$. For NLP tasks, s is a text consisting of words w_i from a dictionary $\mathcal{X}$. Our objective is to craft an adversarial sequence s_{adv} that misleads f_θ to produce an incorrect prediction by replacing as few elements in the input sequence s as possible. Formally, this can be written as the following optimization problem:

$$\begin{aligned} &\underset{s' \in \mathcal{X}^l}{\text{minimize}} \;\; d(s, s') \\ &\text{subject to} \;\; \mathcal{L}(f_\theta(s'), y) \geq 0, \end{aligned} \tag{1}$$

where d is a distance metric that quantifies the amount of perturbation between two sequences (*e.g.*, Hamming distance) and $\mathcal{L}(f_\theta(s), y) \triangleq \max_{y' \in \mathcal{Y}, y' \neq y} f_\theta(s)_{y'} - f_\theta(s)_y$ denotes the attack criterion. In this paper, we consider the score-based black-box attack setting, where an adversary has access to the model prediction logits with a limited query budget, but not the model configurations such as network architectures and parameters.

To make the adversarial perturbation imperceptible to humans, the modified sequence should be semantically similar to the original sequence and the perturbation size should be sufficiently small (Ren et al., 2019). However, minimizing only the perturbation size does not always ensure the semantic similarity between the two sequences. For example, in the NLP domain, even a single word replacement can completely change the meaning of the original text due to the characteristics of natural languages. To address this, we replace elements with ones that are semantically similar to generate an adversarial example, which is a standard practice in the prior works in NLP. Concretely, we first define a set of semantically similar candidates $\mathcal{C}(w_i) \subseteq \mathcal{X}$ for each i-th element w_i in the original sequence. In the NLP domain, this can be found by existing word substitution methods (Ren et al., 2019; Jin et al., 2020; Zang et al., 2020). Then, we find an adversarial sequence in their product space $\prod_{i=0}^{l-1} \mathcal{C}(w_i) \subseteq \mathcal{X}^l$.

We emphasize that the greedy-based attack methods have the restricted search spaces of size $\sum_{i=0}^{l-1} |\mathcal{C}(w_i)| - l + 1$. In contrast, our search space is of cardinality $|\prod_{i=0}^{l-1} \mathcal{C}(w_i)|$, which is always larger than the greedy methods.

3.2. Bayesian Optimization

Bayesian optimization is one of the most powerful approaches for maximizing a black-box function $g : A \rightarrow \mathbb{R}$ (Snoek et al., 2012; Frazier, 2018). It constructs a probabilistic model that approximates the true function g, also referred to as a surrogate model, which can be evaluated relatively cheaply. The surrogate model assigns a prior distribution to g and updates the prior with the evaluation history to get a posterior distribution that better approximates g. Gaussian processes (GPs) are common choices for the surrogate model due to their flexibility and theoretical properties (Osborne et al., 2009). A GP prior assumes that the values of g on any finite collection of points $X \subseteq A$ are normally distributed, *i.e.*, $g(X) \sim \mathcal{N}(\mu(X), K(X, X) + \sigma_n^2 I)$, where $\mu : A \rightarrow \mathbb{R}$ and $K : A \times A \rightarrow \mathbb{R}$ are the mean and kernel functions, respectively, and σ_n^2 is the noise variance. Given the evaluation history $\mathcal{D} = \{(\hat{x}_j, \hat{y}_j = g(\hat{x}_j))\}_{j=0}^{n-1}$, the posterior distribution of g on a finite candidate points X can also be expressed as a Gaussian distribution with the predictive mean and variance as follows:

$$\begin{aligned} &\mathrm{E}[g(X) \mid X, \mathcal{D}] \\ &\quad = K(X, \hat{X})[K(\hat{X}, \hat{X}) + \sigma_n^2 I]^{-1}(\hat{Y} - \mu(\hat{X})) + \mu(X) \\ &\mathrm{Var}[g(X) \mid X, \mathcal{D}] \\ &\quad = K(X, X) - K(X, \hat{X})[K(\hat{X}, \hat{X}) + \sigma_n^2 I]^{-1} K(\hat{X}, X), \end{aligned}$$

where $\hat{X}$ and $\hat{Y}$ are the concatenations of $\hat{x}_j$'s and $\hat{y}_j$'s, respectively.

Based on the current posterior distribution, an acquisition function quantifies the utility of querying g at each point for the purpose of finding the maximizer. Bayesian optimization proceeds by maximizing the acquisition function to determine the next point x_n to evaluate and updating the posterior distribution with the new evaluation history $\mathcal{D} \cup \{(\hat{x}_n, g(\hat{x}_n))\}$. After a fixed number of function evaluations, the point evaluated with the largest $g(x)$ is returned as the solution.

4. Methods

In this section, we introduce the proposed *Blockwise Bayesian Attack* (BBA) framework. Instead of optimizing Equation (1) directly, we divide the optimization into two steps. First, we conduct Bayesian optimization to maximize the black-box function $\mathcal{L}(f_\theta(\cdot), y)$ on the attack space $\mathcal{S} \triangleq \prod_{i=0}^{l-1} \mathcal{C}(w_i)$ until finding an adversarial sequence s_{adv}, which is a feasible solution of Equation (1). This step can be formulated as

$$\underset{s' \in \mathcal{S}}{\text{maximize}} \;\; \mathcal{L}(f_\theta(s'), y). \tag{2}$$

Second, after finding a valid adversarial sequence s_{adv} that satisfies the attack criterion $\mathcal{L}(f_\theta(s_{\text{adv}}), y) \geq 0$, we seek to reduce the Hamming distance of the perturbed sequence from the original input while maintaining the constant feasibility.

Note that Equation (2) is a high-dimensional Bayesian optimization problem on combinatorial search space, especially for datasets consisting of long sequences. However, the number of queries required to obtain good coverage of the input space, which is necessary to find the optimal solution,

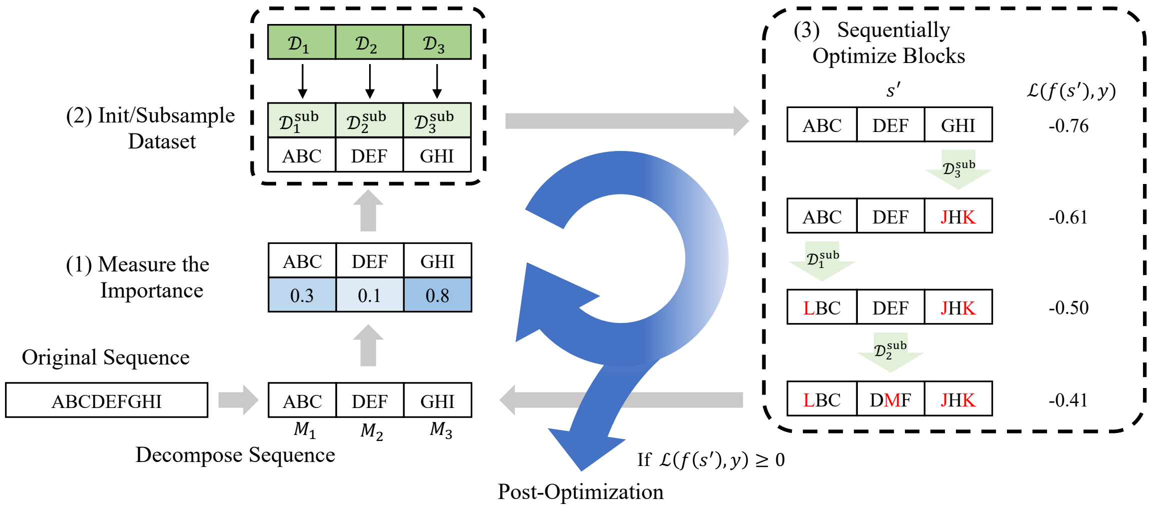

Figure 1: The overall process of BBA. A green arrow with a dataset $\mathcal{D}_k^{\text{sub}}$ denotes the Bayesian optimization step for the block M_k using D_k^{sub} as the initial dataset.

increases exponentially with respect to the input dimensions due to the curse of dimensionality (Shahriari et al., 2015). This high *query complexity* is prohibitive for query-efficient adversarial attacks. Furthermore, even in a low-dimensional space, the high *computational complexity* of training GP models in Bayesian optimization can drastically slow down the runtime of the algorithm as the evaluation history becomes larger. Fitting the GP model requires the matrix inversion of the covariance matrix $K(\hat{X},\hat{X})$, whose computational complexity is $\mathcal{O}(n^3)$, where n is the number of evaluations so far.

To this end, we first introduce the surrogate model and the parameter fitting method which are suitable for our high-dimensional combinatorial search space. Next, we propose two techniques to deal with the scalability issues that arise from the high query and computational complexity of Bayesian optimization. Lastly, we introduce a post-optimization technique that effectively minimizes the perturbation size of an adversarial sequence.

4.1. Surrogate Model and GP Parameter Fitting

Choosing an appropriate kernel that captures the structure of the high-dimensional combinatorial search space is the key to the success of our GP-based surrogate model. We use a categorical kernel[1] with automatic relevance determination (ARD) to automatically determine the degree to which each input dimension is important (MacKay, 1992). The kernel has the following form:

$$K^{\text{cate}}(s^{(1)}, s^{(2)}) = \sigma_f^2 \prod_{i=0}^{l-1} \exp\left(-\frac{\mathbf{1}[w_i^{(1)} \neq w_i^{(2)}]}{\beta_i}\right),$$

where σ_f^2 is a signal variance, β_i is a length-scale parameter corresponding to the relevance of i-th element position. This implies that the kernel regards a sequence pair sharing a larger number of elements as a more similar pair. The GP parameter β_i is estimated by maximizing the posterior probability of the evaluation history under a prior using the gradient descent with Adam optimizer (Kingma & Ba, 2015). More details can be found in Appendix B.

4.2. Techniques for Scalability

To achieve a scalable Bayesian optimization algorithm, we decompose an input sequence into disjoint blocks of element positions and optimize each block in a sequential fashion for several iterations using data subsampled from the evaluation history corresponding to the block.

4.2.1. Block Decomposition

We divide an input sequence of length l into $\lceil l/m \rceil$ disjoint blocks of length m. Each k-th block M_k consists of consecutive indices $[km, \dots, (k+1)m-1]$. We sequentially optimize each block for R iterations, rather than updating all element positions concurrently. For each iteration, we set the maximum query budget to N_k when optimizing the block M_k. While the dimension of the attack space $\prod_{i=0}^{l-1} \mathcal{C}(w_i)$ grows exponentially as l increases, the block decomposition makes the dimension of the search space of each Bayesian optimization step independent of l and upper

[1] https://botorch.org/api/_modules/botorch/models/kernels/categorical.html

Algorithm 1 $\mathrm{SoD}(\mathcal{D}, N)$, Subset of Data method

```
 1: **Input:** The evaluation history $\mathcal{D}$, the size of subsamples $N$.
 2: **if** $|\mathcal{D}| < N$ **then**
 3:     **Return** $\mathcal{D}$.
 4: **end if**
 5: Initialize the dataset $\mathcal{D}_{\text{sub}} \leftarrow \{s_0\}$ where $s_0$ is randomly sampled from $\mathcal{D}$.
 6: **while** $|\mathcal{D}_{\text{sub}}| < N$ **do**
 7:     Select the farthest sequence.
        $s_{\text{far}} \leftarrow \mathrm{argmax}_{s \in \mathcal{D} \setminus \mathcal{D}_{\text{sub}}} [\min_{s' \in \mathcal{D}_{\text{sub}}} d(s, s')]$
 8:     Update the dataset $\mathcal{D}_{\text{sub}} \leftarrow \mathcal{D}_{\text{sub}} \cup \{s_{\text{far}}\}$.
 9: **end while**
10: **Return** $\mathcal{D}_{\text{sub}}$.
```

bounded by $(C_{\max})^m$, where $C_{\max} \triangleq \max_{i \in [l]} |\mathcal{C}_i|$ is the size of the largest synonym set.

At the start of each iteration, we assign an importance score to each block, which measures how much each block contributes to the objective function value. Then, we sequentially optimize blocks in order of highest importance score for query efficiency. For the first iteration, we set the importance score of each block to the change in the objective function value after deleting the block. For the remaining iterations, we reassign the importance score to each block M_k by summing the inverses of the length-scale parameters that correspond to the element positions in M_k, *i.e.*, $\sum_{i \in M_k} 1/\beta_i$.

4.2.2. History Subsampling

Here, we propose a data subsampling strategy suitable for our block decomposition method. When we optimize a block M_k, only the elements in M_k are updated while the remaining elements are unchanged. Thus, in terms of the block M_k, all sequences evaluated during the optimization steps for blocks other than M_k share the same elements, which do not provide any information on how much M_k affects the objective function value. To avoid this redundancy, we consider utilizing only the sequences collected from the previous optimization steps for M_k as the evaluation history, denoted by $\mathcal{D}_k$, when optimizing M_k.

On top of the strategy above, we further reduce the computational complexity of Bayesian optimization by subsampling a dataset from the evaluation history and training the GP surrogate model with the reduced dataset. We adopt the Subset of Data (SoD) method with Farthest Point Clustering (FPC) (Chalupka et al., 2012), a simple and efficient subsampling method widely used in the GP literature. Concretely, we randomly sample an initial sequence from the evaluation history and sequentially select the farthest sequence that maximizes the Hamming distance to the nearest of all sequences picked so far. The overall procedure is shown in Algorithm 1. When optimizing a block M_k at each iteration, we select a subset $\mathcal{D}_k^{\text{sub}}$ from the evaluation history $\mathcal{D}_k$ via the subsampling algorithm above and proceed with the Bayesian optimization step for M_k using $\mathcal{D}_k^{\text{sub}}$ as the initial dataset for the GP model training.

Algorithm 2 $\mathrm{PostOpt}(s, s_{\text{adv}}, \mathcal{D}_{\text{sub}}, N_{\text{post}}, N_b)$

```
 1: **Input:** The original sequence $s$, an adversarial sequence $s_{\text{adv}}$, the evaluation dataset $\mathcal{D}_{\text{sub}}$ subsampled from the evaluation history, the query budget $N_{\text{post}}$, and the batch size $N_b$.
 2: Initialize $N_r \leftarrow N_{\text{post}}$.
 3: **while** $N_r > 0$ **do**
 4:     Fit GP parameters to maximize the posterior probability distribution on $\mathcal{D}_{\text{sub}}$.
 5:     Select a batch $B$ of the size $\min(N_b, N_r)$ from $\mathcal{B}_H(s, d_H(s, s_{\text{adv}}) - 1) \cap \mathcal{B}_H(s_{\text{adv}}, r)$ according to the acquisition function and the DPP.
 6:     Evaluate the batch $\mathcal{D}_{\text{batch}} = \{(s', \mathcal{L}(f_\theta(s'), y)\}_{s' \in B}$.
 7:     Update the dataset $\mathcal{D}_{\text{sub}} \leftarrow \mathcal{D}_{\text{sub}} \cup \mathcal{D}_{\text{batch}}$.
 8:     $N_r \leftarrow N_r - |\mathcal{D}_{\text{batch}}|$.
 9:     **if** $B$ has an adversarial sequence **then**
10:         Update $s_{\text{adv}}$ to the best adversarial sequence in $B$.
11:         $N_r \leftarrow N_{\text{post}}$.
12:     **end if**
13: **end while**
14: **Return** $s_{\text{adv}}$.
```

Here, we simply set the initial subset size to N_k, which is the same as the maximum query budget when optimizing the block M_k. Thus, the size of the dataset $\mathcal{D}_k^{\text{sub}}$ during a single block optimization step is upper bounded by $\mathcal{O}(N_k)$. Therefore, we can write the complexity of the GP model fitting step when optimizing a block by $\mathcal{O}((\max_k N_k)^3)$, which is independent of the total number of evaluations, n. More details containing the runtime analysis of the overall process can be found in Appendices C.1 and C.2.

4.2.3. Acquisition Maximization Considering Batch Diversity via Determinantal Point Process

We utilize expected improvement as the acquisition function, which is defined as $\mathrm{EI}(x) = \mathrm{E}[\max(g(s) - g_{\mathcal{D}}^*, 0)]$, where $g_{\mathcal{D}}^* = \max_{\hat{y} \in \hat{Y}} \hat{y}$ is the largest value evaluated so far.

To further enhance the runtime of the Bayesian optimization algorithm, we evaluate a batch of sequences parallelly in a single round, following the practice in Wang et al. (2017). We sample an evaluation batch B via a Determinantal Point Process (DPP), which promotes batch diversity by maximizing the determinant of its posterior variance matrix $\mathrm{Var}(g(B) \mid \mathcal{D})$ (Kulesza & Taskar, 2012). Concretely, we

Table 1: Attack results for XLNet-base, BERT-base, and LSTM models on sentence-level classification datasets.

(a) WordNet

Dataset	Model	Method	ASR (%)	MR (%)	Qrs
AG	BERT-base	PWWS	57.1	18.3	367
		BBA	**77.4**	**17.8**	**217**
	LSTM	PWWS	78.3	16.4	336
		BBA	**83.2**	**15.4**	**190**
MR	XLNet-base	PWWS	83.9	**14.4**	143
		BBA	**87.8**	**14.4**	**77**
	BERT-base	PWWS	82.0	15.0	143
		BBA	**88.3**	**14.6**	**94**
	LSTM	PWWS	**94.2**	13.3	132
		BBA	**94.2**	**13.0**	**67**

(b) Embedding

Dataset	Model	Method	ASR (%)	MR (%)	Qrs
AG	BERT-base	TF	84.7	24.9	346
		BBA	**96.0**	**18.9**	**154**
	LSTM	TF	94.9	17.3	228
		BBA	**98.5**	**16.6**	**142**
MR	XLNet-base	TF	95.0	18.0	101
		BBA	**96.3**	**16.2**	**68**
	BERT-base	TF	89.2	20.0	115
		BBA	**95.7**	**16.9**	**67**
	LSTM	TF	**98.2**	13.6	72
		BBA	**98.2**	**13.1**	**54**

(c) HowNet

Dataset	Model	Method	ASR (%)	MR (%)	Qrs
AG	BERT-base	PSO	67.2	21.2	65860
		BBA	**70.8**	**15.5**	**5176**
	LSTM	PSO	71.0	19.7	44956
		BBA	**71.9**	**13.7**	**3278**
MR	XLNet-base	PSO	**91.3**	18.6	4504
		BBA	**91.3**	**11.7**	**321**
	BERT-base	PSO	**90.9**	17.3	6299
		BBA	**90.9**	**12.4**	**403**
	LSTM	PSO	**94.4**	15.3	2030
		BBA	**94.4**	**11.2**	**138**

Table 2: Attack results for BERT-base models on document-level classification datasets.

(a) WordNet

Dataset	Method	ASR (%)	MR (%)	Qrs
IMDB	PWWS	97.6	4.5	1672
	BBA	**99.6**	**4.1**	**449**
	LSH	96.3	5.3	557
	BBA	**98.9**	**4.8**	**372**
Yelp	PWWS	94.3	7.6	1036
	BBA	**99.2**	**7.4**	**486**
	LSH	92.6	9.5	389
	BBA	**98.8**	**8.8**	**271**

(b) Embedding

Dataset	Method	ASR (%)	MR (%)	Qrs
IMDB	TF	99.1	8.6	712
	BBA	**99.6**	**6.1**	**339**
	LSH	98.5	5.0	770
	BBA	**99.8**	**4.9**	**413**
Yelp	TF	93.5	11.1	461
	BBA	**99.8**	**9.6**	**319**
	LSH	94.7	8.9	550
	BBA	**99.8**	**8.6**	**403**

(c) HowNet

Dataset	Method	ASR (%)	MR (%)	Qrs
IMDB	PSO	**100.0**	3.8	113343
	BBA	**100.0**	**3.3**	**352**
	LSH	98.7	3.2	640
	BBA	**99.8**	**3.0**	**411**
Yelp	PSO	**98.8**	10.6	86611
	BBA	**98.8**	**8.2**	**283**
	LSH	93.9	8.0	533
	BBA	**98.2**	**7.4**	**353**

first select sequences with top-T acquisition values in the 1-Hamming distance ball $\mathcal{B}_H(s^*_{\mathcal{D}}, 1)$ of the best sequence $s^*_{\mathcal{D}}$ evaluated so far. Then, we greedily choose N_b sequences among the top-T sequences that maximize the determinant. More details can be found in Appendix C.1.

4.3. Post-Optimization for Perturbation Reduction

Since we do not consider the perturbation size during the first step of BBA, we conduct a post-optimization step to reduce the perturbation size. To this end, we optimize for a sequence near the current adversarial sequence s_{adv} that stays adversarial and has a smaller perturbation than s_{adv}. To achieve this, we search an adversarial sequence in a reduced search space $\mathcal{B}_H(s, d_H(s, s_{\text{adv}}) - 1) \cap \mathcal{B}_H(s_{\text{adv}}, r)$, where r controls the exploration size. We also conduct Bayesian optimization for the post-optimization step. We leverage the evaluation history collected during the first step of BBA and subsample an initial dataset for the GP model training from the history. If we find a new adversarial sequence during this step, we replace the current adversarial sequence with the new sequence and repeat the step above until we cannot find a new adversarial sequence using the query budget N_{post} after the most recent update. The overall post-optimization procedure is summarized in Algorithm 2.

Figure 1 illustrates the overall process of BBA. Please refer to Algorithm 3 in Appendix A for the more detailed overall algorithm of BBA.

5. Experiments

We evaluate the performance of BBA on text classification, textual entailment, and protein classification tasks. We first provide a brief description of the datasets, victim models, and baseline methods used in the experiments. Then, we report the performance of BBA compared to the baselines. Our implementation is available at `https://github.com/snu-mllab/DiscreteBlockBayesAttack`.

5.1. Datasets and Victim Models

To demonstrate the wide applicability and effectiveness of BBA, we conduct experiments on various datasets in the NLP and protein domain. In the NLP domain, we use sentence-level text classification datasets (AG's News, Movie Review), document-level classification datasets (IMDB, Yelp), and textual entailment datasets (MNLI, QNLI) (Zhang et al., 2015; Pang & Lee, 2005; Maas et al., 2011; Williams et al., 2018; Wang et al., 2018a). In the protein domain, we use an enzyme classification dataset (EC) with 3-level hierarchical multi-labels (Strodthoff et al., 2020). Note that a protein is a sequence of amino acids, each of which is a discrete categorical variable.

We consider multiple types of victim models to attack, including bi-directional word LSTM, ASGD Weight-Dropped LSTM, fine-tuned BERT-base and BERT-large, and fine-tuned XLNet-base and XLNet-large (Hochreiter & Schmidhuber, 1997; Merity et al., 2018; Devlin et al., 2019; Yang

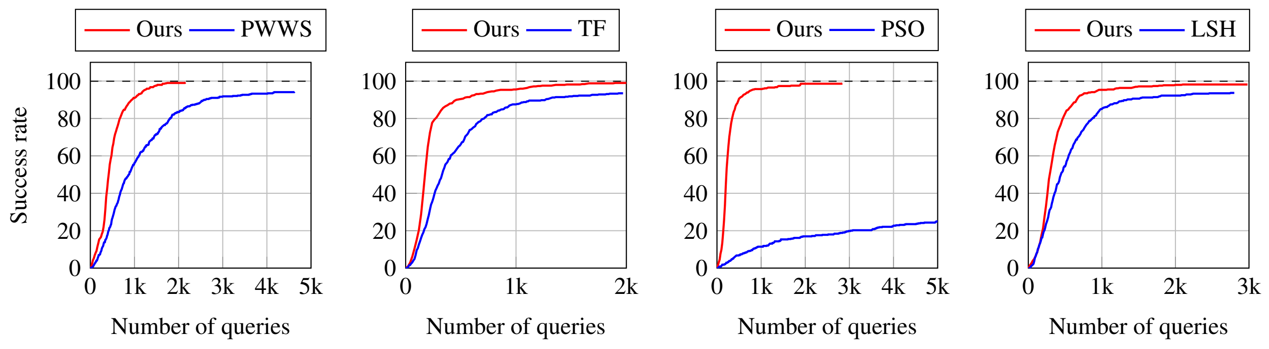

Figure 2: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the Yelp dataset. We use the HowNet based word substitution when comparing our method against LSH.

et al., 2019). More details on datasets and victim models can be found in Appendices C.3 and C.4, respectively.

5.2. Baseline Methods

In the NLP domain, we compare the performance of BBA against the state-of-the-art methods such as PWWS, TextFooler, LSH, BAE, and PSO, the first four of which are greedy-based algorithms (Ren et al., 2019; Jin et al., 2020; Maheshwary et al., 2021; Garg & Ramakrishnan, 2020; Zang et al., 2020). Note that PWWS, TextFooler, BAE, and PSO have different attack search spaces since they utilize different word substitution methods (WordNet, Embedding, BERT masked language model, and HowNet, respectively) (Fellbaum, 1998; Mrkšić et al., 2016; Dong et al., 2010). For a fair comparison, we follow the practice in Maheshwary et al. (2021) and compare BBA against each baseline individually under the same attack setting (*e.g.*, word substitution method, query budget) as used in the baseline. We also note that LSH leverages additional attention models, each of which is pre-trained on a different classification dataset. Please refer to Appendix C.5 for more details.

For the protein classification task, we compare BBA with TextFooler. To define its attack space, we exploit the experimental exchangeability of amino acids (Yampolsky & Stoltzfus, 2005), which quantifies the mean effect of exchanging one amino acid to a different amino acid on protein activity, as the measure of semantic similarity. Then, we define a synonym set for each amino acid by thresholding amino acids with the experimental exchangeability and set the attack space to the product of the synonym sets. As in the NLP domain, we compare BBA with the baseline under the same experimental setting as used in the baseline.

5.3. Attack Performance

We quantify the attack performance in terms of three main metrics: attack success rate (ASR), modification rate (MR), and the average number of queries (Qrs). The attack success rate is defined as the rate of successfully finding misclassified sequences from the original sequences that are correctly classified, which directly measures the effectiveness of the attack method. The modification rate is defined as the percentage of modified elements after the attack, averaged over successfully fooled sequences. This rate is formally written by $\mathrm{E}[d_H(s, s_{\text{adv}})/\mathrm{len}(s)]$, which quantifies the distortion of the perturbed sequences from the original. The average number of queries, computed over all sequences being attacked, represents the query efficiency of the attack methods.

The main attack results on text classification tasks are summarized in Tables 1 and 2. The results show that BBA significantly outperforms all the baseline methods in all the evaluation metrics for all datasets and victim models we consider. Figure 2 shows the cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the Yelp dataset. The results show that BBA finds successful adversarial texts using less number of queries than the baseline methods. More experimental results on other target models (BERT-large, XLNet-large), baseline method (BAE), and datasets (MNLI, QNLI) can be found in Appendix D.1.

Moreover, Table 3 shows that BBA outperforms the baseline method by a large margin for the protein classification task, which shows the general applicability and effectiveness of BBA on multiple domains.

Table 3: Attack results against AWD-LSTM models on the protein classification dataset EC50 level 0, 1, and 2.

	Level 0			Level 1			Level 2		
Method	ASR	MR	Qrs	ASR	MR	Qrs	ASR	MR	Qrs
TF	83.8	3.2	619	85.8	3.0	584	89.6	2.5	538
BBA	**99.8**	**2.9**	**285**	**99.8**	**2.3**	**293**	**100.0**	**2.0**	**231**

For a direct comparison with a baseline, one can compute the MR and Qrs over the texts that both BBA and the baseline method are successful on. Table 4 shows that BBA outperforms PWWS in MR and Qrs on samples that both methods successfully fooled by a larger margin.[2]

Table 4: Attack results on the AG's News. MR and Qrs of 'both success' are averaged over the texts that both PWWS and BBA successfully fooled.

					Both success	
Model	Method	ASR (%)	MR (%)	Qrs	MR (%)	Qrs
BERT-base	PWWS	57.1	18.3	367	17.8	311
	BBA	**77.4**	**17.8**	**217**	**14.0**	**154**
LSTM	PWWS	78.3	16.4	336	16.1	311
	BBA	**83.2**	**15.4**	**190**	**14.4**	**163**

5.4. Ablation Studies

5.4.1. The Effect of DPP in Batch Update

To validate the effectiveness of the DPP-based batch update technique, we compare BBA with the greedy-style batch update which chooses the sequences of top-N_b acquisition values for the next evaluations. We do not utilize the post-optimization process to isolate the effect of the batch update. Table 5 shows that the batch update with DPP consistently achieves higher attack success rate using fewer queries compared to the greedy-style batch update. Surprisingly, the batch update with DPP achieves higher attack success rate using fewer queries compared to 'without batch update' in AG's News dataset.

Table 5: Attack results of BBA with and without batch update using the WordNet-based word substitution against BERT-base and LSTM models on the sentence-level classification datasets. 'Top-N_b' denotes the greedy-style batch update that chooses sequences of top-N_b acquisition values.

		BERT-base		LSTM	
Dataset	Method	ASR (%)	Qrs	ASR (%)	Qrs
AG	w/o batch	76.1	126	73.5	127
	w/ batch, Top-N_b	75.9	133	74.3	127
	w/ batch, DPP	**77.4**	**124**	**83.2**	**86**
MR	w/o batch	88.5	26	93.9	18
	w/ batch, Top-N_b	87.1	28	93.6	20
	w/ batch, DPP	**88.3**	**25**	**94.2**	**17**

5.4.2. The Effect of Post-Optimization Process

We analyze the trend of change in modification rate during the post-optimization process. The post-optimization process reduces the distortion between the adversarial sequence and the original sequence using additional queries, which results in a trade-off between the distortion and the number of queries. Figure 3 shows the trajectory of the modification rate while traversing the query budget N_{post} for post-optimization from 0 to 200. We find that the post-optimization process reduces the distortion and reaches the same distortion as PWWS using a fewer number of queries.

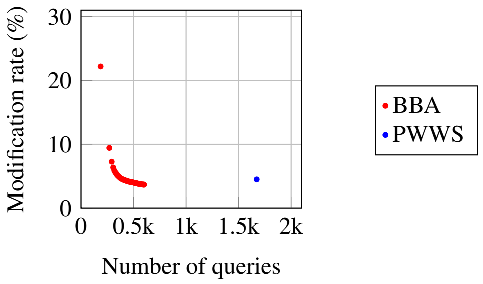

Figure 3: Modification rate versus the number of queries plot of adversarial texts generated by traversing N_{post} from 0 to 200 on the IMDB dataset against a BERT-base model. We use WordNet substitution method for the attack.

5.4.3. The Actual Runtime Analysis

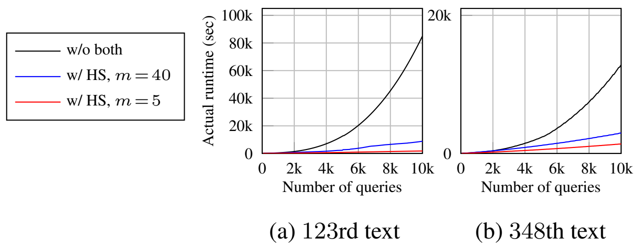

Figure 4: The cumulative runtime versus the number of queries plot. HS in the legend denotes history subsampling, and $m = k$ in the legend denotes block decomposition with the block size k.

To study the effectiveness of block decomposition and history subsampling techinques on runtime, we choose two texts (123rd text of length 641 and 348th text of length 40) from the texts that BBA iterates more than once until attack success when attacking the Yelp dataset against BERT-base model. Figure 4 shows that block decomposition and history subsampling significantly reduces the actual runtime as the number of queries increases. Note that the comparison between the black and the blue curve of 348th text shows only the effect of history subsampling since the block size is equal to the text length (single block). In practice, attacking long documents against the robust model may require a large number of queries and our techniques can effectively

[2]For BERT-base on AG, PWWS fools 267 texts, BBA fools 363 texts, and both commonly fools 262 texts among 500 texts. For LSTM on AG, PWWS fools 354 texts, BBA fools 376 texts, and both commonly fools 349 texts among 500 texts.

Table 6: Examples of the original and their adversarial sequences from Yelp and EC50 against BERT-base models.

Document-Level Text Classification (Yelp)		Label
Orig	Food is fantastic and exceptionally clean! My only complaint is I went there with my 2 small children and they were showing a very inappropriate R rated movie!	Positive
BBA	Food is *gorgeous* and exceptionally *unpolluted*! My only complaint is I went there with my 2 small children and they were showing a very inappropriate R rated movie!	Negative
TF	Food is fantastic and *awfully* clean! My only *grievances* is I *turned* there with my 2 small children and they were showing a very inappropriate R rated *footage*!	Negative
Protein Classification (EC50 level 0)		**Label**
Orig	`MATPWRRALLMILASQVVTLVKCLEDDDVPEEWLLLHVVQGQIGAGNYSYLRLNHEGKIILRMQSLRGDADLYVSDSTPHPSFDDYELQSVT` `CGQDVVSIPAHFQRPVGIGIYGHPSHHESDFEMRVYYDRTVDQYPFGEAAYFTDPTGASQQQAYAPEEAAQEEESVLWTILISILKLVLEILF`	Non-Enzyme
BBA	`MATPWRRALLMRLASQVVTLVKCLEDDDVPEEWLLLHVVQGQIGAGNYSYLRLNHEGKIILRMQSLRGDADLYVSDSTPHPSFDDYELQSVT` `CGQDVVSIPAHFQRPVGIGIYGHPSHHESDFEMRVYYDWTVDWYPFGEAAYFTDPTGASQQQAYAPEEAAQEEESVLWTILISILKLVLEILF`	Enzyme
TF	`MATPWRRALLMILASQVVTLVKCLEDDDVPEEWLLLHVVQGQIGAGNYSYLRLNHEGKIILRMQSLRGDADLYVSDSTPHPSFDDYELQSVT` `CGQDVVSIPAHFQRPVGIGIYGHPSHHESDFEMRVYYDRTVDQYPFGEWAYFCCGWGASQQQAYAPEEWWWFEESVLDTILISGLKLVLEILF`	Enzyme

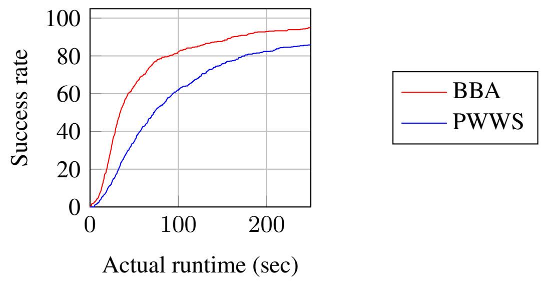

Figure 5: The cumulative distribution of the actual runtime required for the attack methods against the XLNet-large model on the Yelp dataset. Refer to Table 16 in Appendix D.1 for the detailed attack results.

reduce the actual runtime in that situation.

Figure 5 shows the cumulative distribution of the actual runtime required for attack methods. The result shows that BBA consistently finds successful texts faster than PWWS against the XLNet-large model on the Yelp dataset. Note that one could further accelerate the kernel computations of Bayesian optimization using a better computation resource such as a multi-GPU cluster.

5.5. Qualitative Results

Attack examples of Yelp and EC50 datasets in Table 6 show that our method successfully generates semantically consistent adversarial texts while baseline methods generate adversarial sequences with high modification rates. Please refer to Appendix D.4 for more qualitative results.

6. Conclusion

We propose a query-efficient and scalable black-box attack method on discrete sequential data using Bayesian optimization. In contrast to greedy-based state-of-the-art methods, our method can dynamically compute important positions using an ARD categorical kernel during Bayesian optimization. Furthermore, we propose block decomposition and history subsampling techniques to scale our method to long sequences and large queries. Lastly, we develop a post-optimization algorithm that minimizes the perturbation size. Our extensive experiments on various victim models and datasets from different domains show the state-of-the-art attack performance compared to the baseline methods. Our method achieves a higher attack success rate with a significantly lower modification rate and the number of queries throughout all our experiments.

Broader Ethical Impact

Our research focuses on the important problem of adversarial vulnerabilities of classification models on discrete sequential data. Even though there is the possibility of a malicious adversary misusing BBA to attack public text classification APIs, we believe our research can be a basis for the improvement in defenses against adversarial attacks on discrete sequential data.

Acknowledgements

This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2101-01, Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00882, (SW STAR LAB) Development of deployable learning intelligence via self-sustainable and trustworthy machine learning), and Basic Science Research Program through the National Research Foundation of Korea (NRF) (2020R1A2B5B03095585). This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-20-1-4043. Hyun Oh Song is the corresponding author.

References

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., and Chang, K.-W. Generating natural language adversarial examples. In *EMNLP*, 2018.

Amazon Comprehend. https://aws.amazon.com/comprehend, 2022.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV*, 2020.

Azimi, J., Fern, A., and Fern, X. Z. Batch bayesian optimization via simulation matching. In *NeurIPS*, 2010.

Bairoch, A. and Apweiler, R. The SWISS-PROT Protein Sequence Data Bank and Its New Supplement TREMBL. *Nucleic Acids Research*, 24(1):21–25, 1996.

Baptista, R. and Poloczek, M. Bayesian optimization of combinatorial structures. In *ICML*, 2018.

Chalupka, K., Williams, C., and Murray, I. A framework for evaluating approximation methods for gaussian process regression. In *JMLR*, 2012.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Dong, Z., Dong, Q., and Hao, C. HowNet and its computation of meaning. In *Coling*, 2010.

Eriksson, D. and Jankowiak, M. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *UAI*, 2021.

Fellbaum, C. Wordnet: An electronic lexical database. In *Bradford Books*, 1998.

Frazier, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Garg, S. and Ramakrishnan, G. BAE: BERT-based adversarial examples for text classification. In *EMNLP*, 2020.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38: 293–306, 1985.

Google Cloud NLP. https://cloud.google.com/natural-language, 2022.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–1780, 1997.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In *ICML*, 2018.

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

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional bayesian optimisation and bandits via additive models. In *ICML*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kulesza, A. and Taskar, B. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

Li, L., Ma, R., Guo, Q., Xue, X., and Qiu, X. BERT-ATTACK: Adversarial attack against BERT using BERT. In *EMNLP*, 2020.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, 2011.

MacKay, D. J. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Maheshwary, R., Maheshwary, S., and Pudi, V. A strong baseline for query efficient attacks in a black box setting. In *EMNLP*, 2021.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing LSTM language models. In *ICLR*, 2018.

Morris, J., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *EMNLP*, 2020.

Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Rojas-Barahona, L., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*, 2016.

Oh, C., Tomczak, J., Gavves, E., and Welling, M. Combinatorial bayesian optimization using the graph cartesian product. In *NeurIPS*, 2019.

Osborne, M. A., Garnett, R., and Roberts, S. J. Gaussian processes for global optimization. In *LION3*, 2009.

Pang, B. and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.

Papernot, N., McDaniel, P., Swami, A., and Harang, R. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM*, 2016.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Ren, S., Deng, Y., He, K., and Che, W. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, 2019.

Ru, B., Cobb, A., Blaas, A., and Gal, Y. Bayesopt adversarial attack. In *ICLR*, 2020.

Seeger, M. W., Williams, C. K., and Lawrence, N. D. Fast forward selection to speed up sparse gaussian process regression. In *AISTATS*, 2003.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015.

Shukla, S. N., Sahu, A. K., Willmott, D., and Kolter, J. Z. Black-box adversarial attacks with bayesian optimization. *arXiv preprint arXiv:1909.13857*, 2019.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, 2012.

Strodthoff, N., Wagner, P., Wenzel, M., and Samek, W. UDSMProt: universal deep sequence models for protein classification. *Bioinformatics*, 36(8):2401–2409, 2020.

Wan, X., Kenlay, H., Ru, B., Blaas, A., Osborne, M., and Dong, X. Adversarial attacks on graph classifiers via bayesian optimisation. In *NeurIPS*, 2021.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP*, 2018a.

Wang, Z., Li, C., Jegelka, S., and Kohli, P. Batched high-dimensional bayesian optimization via structural kernel learning. In *ICML*, 2017.

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale bayesian optimization in high-dimensional spaces. In *AISTATS*, 2018b.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020.

Yampolsky, L. Y. and Stoltzfus, A. The exchangeability of amino acids in proteins. *Genetics*, 170(4):1459–1472, 2005.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Yoo, J. Y., Morris, J. X., Lifland, E., and Qi, Y. Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples. *arXiv preprint arXiv:2009.06368*, 2020.

Zang, Y., Yang, C., Qi, F., Liu, Z., Zhang, M., Liu, Q., and Sun, M. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, 2020.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *NeurIPS*, 2015.

A. Algorithms

The overall algorithm of BBA is shown in Algorithm 3. For the ease of notation, we assume that l is divisible by m.

Notations used in Algorithm 3	
$\mathcal{D}_k$	The evaluation history corresponding to the block M_k.
$\mathcal{D}_k^{\text{sub}}$	The evaluation dataset subsampled from the evaluation history.
E_k	Exploration budget when optimizing the block M_k.
N_k	Query budget when optimizing the block M_k.
N_{post}	Query budget for the post-optimization process.
N_b	The batch size.
$\mathcal{S}(s', M_k)$	The attack space when optimizing the block M_k with an initial sequence s'. Formally, $\mathcal{S}(s', M_k) = \{s'' \in \prod_{i=0}^{l-1} \mathcal{C}(w_i') \mid w_i'' = w_i' \text{ for } i \notin M_k\}$.
$s_{\setminus M_k}$	The subsequence of s corresponding to the index set $[l] \setminus M_k$. Formally, $s_{\setminus M_k} = (w_i)_{i \in [l] \setminus M_k}$.

Algorithm 3 The overall algorithm of BBA.

```
1: Input: The original sequence $s$, its label $y$, blocks $\{M_k\}$, a target classifier $f_\theta$ and the attack criterion $\mathcal{L}$.
2: Initialize the current sequence $s_{\text{cur}} \leftarrow s$.
3: Initialize the evaluation history $\mathcal{D}_k \leftarrow \emptyset$ for all $k = 0, 1, \ldots, l/m - 1$.
4: Initialize the importance score $\alpha_k \leftarrow |\mathcal{L}(f_\theta(s), y) - \mathcal{L}(f_\theta(s_{\setminus M_k}), y)|$ for all $k = 0, 1, \ldots, l/m - 1$.
5: for ITER $= 0$ to $R - 1$ do
6:   Sort the block indices $[0, \ldots, l/m - 1]$ to $[\gamma_0, \ldots, \gamma_{l/m-1}]$ in order of decreasing importance score.
7:   for $k$ in $[\gamma_0, \ldots, \gamma_{l/m-1}]$ do
8:     Initialize the evaluation dataset $\mathcal{D}_k^{\text{sub}} \leftarrow \text{SoD}(\mathcal{D}_k, N_k)$ (See Algorithm 1)
9:     Evaluate $E_k$ sequences randomly sampled from the attack space $\mathcal{S}(s_{\text{cur}}, M_k)$ and append them to $\mathcal{D}_k^{\text{sub}}$ and $\mathcal{D}_k$.
10:    Update the remaining budget $N_r \leftarrow N_k - E_k$.
11:    while $N_r > 0$ do
12:      Update $s_{\text{cur}}$ to the best sequence evaluated so far.
13:      if $\mathcal{L}(f_\theta(s_{\text{cur}}), y) \geq 0$ then
14:        Return $\text{PostOpt}(s, s_{\text{cur}}, \text{SoD}(\mathcal{D}_k, N_k), N_{\text{post}}, N_b)$. (See Algorithm 2)
15:      end if
16:      Fit the GP parameters on $\mathcal{D}_k^{\text{sub}}$. (See Appendix B)
17:      Select the sequence set $\mathcal{T}$ of top-$T$ acquisition value in $\mathcal{B}_H(s_{\text{cur}}, 1) \cap \mathcal{S}(s_{\text{cur}}, M_k)$.
18:      Greedily select a batch $B$ of size $\min(N_b, N_r)$ from $\mathcal{T}$ that maximizes its DPP prior. (See Appendix C.1.2)
19:      Evaluate the batch $B$ and append them to $\mathcal{D}_k^{\text{sub}}$ and $\mathcal{D}_k$.
20:      Update the remaining budget $N_r \leftarrow N_r - \min(N_b, N_r)$.
21:    end while
22:  end for
23:  Fit the GP parameters on $\bigcup_k \text{SoD}(\mathcal{D}_k, N_k)$.
24:  Update the importance score $\alpha_k \leftarrow \sum_{i \in M_k} 1/\beta_i$.
25: end for
26: Return The best sequence $s_{\text{best}}$ in $\bigcup_k \mathcal{D}_k$.
```

B. GP Surrogate Model Fitting

We use a GP surrogate model consisting of a constant mean function $\mu(\cdot;\eta)$ with mean η, an ARD categorical kernel $k^{\text{cate}}(\cdot,\cdot;\{\beta_i\},\sigma_f^2)$ with length-scale parameters $\{\beta_i\}$ and signal variance σ_f^2, and a homoskedastic noise with variance σ_n^2. Concretely, the surrogate model g can be written by

$$g(X) \sim \mathcal{N}(\eta, K^{\text{cate}}(X,X;\{\beta_i\},\sigma_f^2) + \sigma_n^2 I).$$

For a given dataset $\mathcal{D}^{\text{sub}}$, we train the parameters η, $\{\beta_i\}$, σ_f^2, and σ_n^2 by maximizing the posterior probability distribution $p(\eta,\{\beta_i\},\sigma_f^2,\sigma_n^2 \mid \mathcal{D}^{\text{sub}})$. Since

$$\overbrace{p(\eta,\{\beta_i\},\sigma_f^2,\sigma_n^2 \mid \mathcal{D}^{\text{sub}})}^{\text{Posterior probability}} \propto p(\eta)\left(\prod_i p(\beta_i)\right) p(\sigma_f^2)p(\sigma_n^2)\underbrace{p(\mathcal{D}^{\text{sub}}|\eta,\{\beta_i\},\sigma_f^2,\sigma_n^2)}_{\text{Marginal likelihood}},$$

we maximize the sum of the log marginal likelihood of the dataset $\mathcal{D}^{\text{sub}}$ and the log prior probabilities of parameters η, $\{\beta_i\}$, σ_f^2, and σ_n^2. Concretely, we estimate η, $\{\beta_i\}$, σ_f^2, and σ_n^2 by solving

$$\underset{\eta,\{\beta_i\},\sigma_f^2,\sigma_n^2}{\text{maximize}} \;\log\left(p\left(\mathcal{D}^{\text{sub}} \mid \eta,\{\beta_i\},\sigma_f^2,\sigma_n^2\right)\right) + \log\left(p(\eta)\right) + \sum_i \log\left(p\left(\beta_i\right)\right) + \log\left(p\left(\sigma_f^2\right)\right) + \log\left(p\left(\sigma_n^2\right)\right). \quad (3)$$

For a more detailed explanation, we first introduce the prior distributions used in our experiments and explain the optimization procedure of Equation (3).

B.1. Prior on Parameters

Here, we provide details about the prior distributions of GP parameters. We assume that the mean η and the signal variance σ_f^2 have uniform priors on their domains. We assign an inverse gamma prior to the length-scale parameter β_i: $\beta_i^{-1} \sim \text{Gamma}(3.0, 6.0)$, where $\text{Gamma}(a,b)$ is a gamma distribution with a concentration parameter a and a rate parameter b. To induce the noise variance σ_n^2 to be close to zero, we use a gamma distribution with low concentration and high rate as a prior: $\sigma_n^2 \sim \text{Gamma}(0.9, 10.0)$.

B.2. Optimization Method

Since the log prior probabilities of GP parameters and log marginal likelihood are differentiable with respect to GP parameters, we use Adam, a gradient-based optimizer, to solve Equation (3) (Kingma & Ba, 2015). For each GP parameter fitting step, we run Adam with a learning rate of 0.1 for 3 iterations starting from the GP parameter derived at the previous fitting step (warm start). Then, we update the GP parameters with the values of the last iteration.

C. Implementation Details

C.1. Experimental Details

C.1.1. Block Decomposition and History Subsampling

As illustrated in Figure 1, we subsample datasets for all blocks before starting the sequential block optimization step. When the optimization step ends, we fit the GP parameters to maximize the posterior probability distribution on $\bigcup_k \text{SoD}(\mathcal{D}_k, N_k)$ and reassign the importance score of the block M_k to $\sum_{i\in M_k} 1/\beta_i$.

For the experiments in the NLP domain, we fix the block size $m = 40$. For the experiments in protein domain, we fix the block size $m = 20$. To allocate a larger query budget to the blocks with larger search space, we fix $N_k = \sum_{i\in M_k}(|\mathcal{C}_i| - 1)$ in all experiments.

C.1.2. Acquisition Maximization

Here, we illustrate our batch update step in detail. We denote the evaluation dataset by $\mathcal{D}$ for notational simplicity. For each acquisition maximization step, we first find the sequence set $\mathcal{T}$ of top-T acquisition values in the 1-Hamming distance ball $\mathcal{B}_H(s_{\mathcal{D}}^*, 1)$ of the best sequence $s_{\mathcal{D}}^*$ evaluated so far. Then, we initialize a batch $B = \{s_{\mathcal{T}}^*\}$, where $s_{\mathcal{T}}^*$ is the sequence with the highest acquisition value in $\mathcal{T}$. Finally, we repeatedly append the sequence s^* that maximizes the marginal gain in the DPP prior to B until the size of B reaches N_b, which is represented as

$$s^* = \underset{s'\in\mathcal{T}\setminus B}{\text{argmax}}\big(\det\left(\text{Var}\left(g(B\cup s' \mid \mathcal{D})\right)\right) - \det\left(\text{Var}\left(g(B \mid \mathcal{D})\right)\right)\big).$$

To avoid redundant evaluations, we exclude sequences previously evaluated in the past acquisition maximization steps from the batch. We fix $N_b = 4$ and $T = 100$ for all experiments.

C.1.3. Post-Optimization

In the post-optimization process, we search an adversarial sequence in a reduced search space $\mathcal{B}_H(s, d_H(s, s_{\text{adv}})-1) \cap \mathcal{B}_H(s_{\text{adv}}, r)$ to reduce the perturbation size. We fix $r = 2$ for all experiments. We choose the next sequence to evaluate by the following steps for efficiency. First, we randomly sample 300 sequences from $\mathcal{B}_H(s_{\text{adv}}, 2) \cap \mathcal{B}_H(s, d_H(s, s_{\text{adv}}) - 2)$ and find the sequence s_{sampled}^* that has the largest acquisition value among them. Then, we choose the sequence $s^* \in \mathcal{B}_H(s_{\text{sampled}}^*, 1)$ that has the largest acquisition value in the 1-Hamming distance ball of s_{sampled}^*.

Since the cardinality of the 1-Hamming distance ball of a sequence is linear to l, we evaluate the acquisition function $\mathcal{O}(l)$ times for each acquisition maximization step. Note that the brute-force acquisition maximization requires $\mathcal{O}(l^2)$ number of the acquisition function evaluations since the search space has the size quadratic to l.

Tables 7 to 11 show the values for the query budget of the post-optimization step N_{post} used in our experiments.

Table 7: N_{post} of BBA used in Table 1 (AG's News, MR).

	$\mathcal{C}$	WordNet	Embedding	HowNet
Dataset	Model	PWWS	TF	PSO
AG	BERT-base	50	20	100
	LSTM	50	20	100
MR	XLNet-base	50	20	100
	BERT-base	100	20	100
	LSTM	50	20	100

Table 8: N_{post} of BBA used in Table 2 (IMDB, Yelp).

	$\mathcal{C}$	WordNet		Embedding		HowNet	
Dataset	Model	PWWS	LSH	TF	LSH	PSO	LSH
IMDB	BERT-base	100	50	20	50	50	100
Yelp	BERT-base	200	50	20	100	50	100

Table 9: N_{post} of BBA used in Table 16 (XLNet-large, BERT-large).

	$\mathcal{C}$	WordNet	Embedding
Dataset	Model	PWWS	TF
IMDB	XLNet-large	100	50
	BERT-large	150	50
Yelp	XLNet-large	100	50
	BERT-large	200	50
MR	XLNet-large	100	20
	BERT-large	200	20

Table 10: N_{post} of BBA used in Table 17 (MNLI, QNLI).

	$\mathcal{C}$	WordNet	Embedding	HowNet
Dataset	Model	PWWS	TF	PSO
MNLI	BERT-base	150	50	150
QNLI	BERT-base	150	50	150

Table 11: N_{post} of BBA used in Table 18 (BAE).

Dataset	MR	AG	Yelp	IMDB
N_{post}	20	20	20	20

For all experiments for the protein classfication task, we use $N_{\text{post}} = 50$.

C.2. Runtime Analysis

In this section, we explain the computational complexity of BBA. We first reiterate the computational complexity of the GP surrogate model fitting and the acquisition maximization in a generic Bayesian optimization. Then we analyze the complexity of each component of BBA. For simplicity, we assume that $|\mathcal{C}(w_i)| = C$ for all i and $N_k = N$ for all k.

As we noted in our main paper, GP surrogate model fitting requires the matrix inversion of the covariance matrix whose computational complexity is $\mathcal{O}(n_{\text{data}}^3)$, where n_{data} is the cardinality of the dataset used in Bayesian optimization. When the inverse matrix of the covariance matrix is given, we can calculate the predictive mean and variance of a sequence by matrix-vector multiplications (see Section 3.2), whose computational complexity is $\mathcal{O}(n_{\text{data}}^2)$. Thus, the computational complexity of the acquisition maximization step is $\mathcal{O}(n_{\text{data}}^2 N_{\text{acq}})$, where N_{acq} is the number of acquisition function queries in the acquisition maximization step.

C.2.1. Sequential Block Optimization

We note that n_{data} is upper bounded by $O(N)$ when optimizing the block M_k since we optimize M_k with an initial dataset of size N and maximum query budget N. Hence, the GP surrogate model fitting step when optimizing M_k requires $\mathcal{O}(N^3)$ computations. The number of the acquisition function queries is $\mathcal{O}(Cm)$ since we find the next sequence to evaluate by maximizing the acquisition function on the 1-Hamming distance ball of the current best sequence. In summary, the sequential block optimization step requires $\mathcal{O}(N^3 + N^2Cm)$ computations for each query, which is independent of n.

C.2.2. History Subsampling

We use the Subset of Data (SoD) method with Farthest Point Clustering (FPC) to select the evaluation dataset $\mathcal{D}_k^{\text{sub}}$ of size N from the evaluation history $\mathcal{D}_k$. Algorithm 1 shows that SoD requires $\mathcal{O}(N|D_k^{\text{sub}}|)$ number of distance computations, resulting in $\mathcal{O}(Nn/m)$ computational complexity. Hence, the history subsampling for all blocks has computational complexity of $\mathcal{O}(Nn)$, which is linear to n.

C.2.3. Importance Score Update

When the optimization step ends, we fit the GP parameters using $\bigcup_k \text{SoD}(\mathcal{D}_k, N)$, whose cardinality is $\mathcal{O}(Nl/m)$. Hence, the importance score update requires $\mathcal{O}(Nn + N^3l^3/m^3)$ computations, where each term corresponds to the history subsampling and the GP surrogate model fitting. Hence, the complexity of the importance score update is linear to n.

C.2.4. Post-Optimization

In the post-optimization process, we search an adversarial sequence in the reduced attack space to minimize the perturbation size. In the worst case, the number of queries required in the post-optimization process is $N_{\text{post}} d_H(s, s_{\text{adv}}^{(\text{init})})$ which is upper bounded by $N_{\text{post}} l$. Here, $s_{\text{adv}}^{(\text{init})}$ denotes the initial adversarial sequence of the post-optimization. Thus, n_{data} is upper bounded by $O(Nm + N_{\text{post}} l)$. Also, N_{acq} is independent of n as explained in Appendix C.1.3. In summary, the computational complexity of the post-optimization process for each query is independent of n.

C.3. Target Datasets

We evaluate BBA on 500 samples randomly selected from the test set throughout all the experiments, following the protocol in Maheshwary et al. (2021).

C.3.1. Text Classification

To show the wide applicability of BBA, we evaluate BBA on various text classification datasets, including both sentence and document-level datasets.

• **AG's News** (Zhang et al., 2015): a sentence-level dataset for classifying a news-type sentence into 4 topics: World, Sports, Business, and Science & Tech.
• **Movie Review** (Pang & Lee, 2005): a sentence-level sentiment classification dataset which consists of positive and negative movie reviews from Rotten Tomatoes.
• **IMDB Polarity** (Maas et al., 2011): a document-level dataset for binary sentiment classification, which consists of highly polar movie reviews from IMDB.
• **Yelp Polarity** (Zhang et al., 2015): a document-level dataset for binary sentiment classification, which consists of highly polar restaurant reviews.

C.3.2. Textual Entailment

We also conduct experiments on textual entailment datasets.

• **MNLI matched** (Williams et al., 2018): a textual entailment dataset, which consists of sentence pairs from transcribed speech, popular fiction, and government reports. The task is to judge the relationship between a pair of sentences, premise, and hypothesis. The test and training sets are derived from the same sources.
• **QNLI** (Wang et al., 2018a): a textual entailment dataset, which consists of question-sentence pairs converted from SQuAD v1.1 dataset (Rajpurkar et al., 2016). The task is to determine whether the sentence contains the answer to the given question or not.

Note that we only perturb hypotheses in MNLI and sentences in QNLI.

Symbol	Amino acid
A	Alanine
R	Arginine
N	Asparagine
D	Aspartic acid
C	Cysteine
Q	Glutamine
E	Glutamic acid
G	Glycine
H	Histidine
I	Isoleucine
L	Leucine
K	Lysine
M	Methionine
F	Phenylalanine
P	Proline
O	Pyrrolysine
S	Serine
U	Selenocysteine
T	Threonine
W	Tryptophan
Y	Tyrosine
V	Valine
B	Aspartic acid or Asparagine
Z	Glutamic acid or Glutamine
X	Any amino acid
__bos__	Beginning of a sentence (BOS) token
__mask__	Mask token
__pad__	Pad token

Table 12: The description of the 28 symbols used in EC50 dataset.

C.3.3. Protein Classification

Furthermore, we consider a protein classification dataset to show that BBA is also applicable to tasks other than NLP tasks. We note that a protein is a sequence of amino acids, each of which is a discrete categorical variable.

• **EC50** (Strodthoff et al., 2020): an enzyme multi-label classification dataset generated by clustering amino acid sequences in Swiss-Prot database (Bairoch & Apweiler, 1996). Each amino acid sequence in EC50 has three hierarchical labels, which correspond to enzyme versus non-enzyme (level 0, 2 classes), main enzyme class (level 1, 6 classes), and enzyme subclass (level 2, 65 classes), respectively.

Each sequence in EC50 is coded with 28 symbols consisting of 25 amino acids from Strodthoff et al. (2020) and 3 auxiliary tokens. The symbols are summarized in Table 12.

C.4. Victim Models

For the sentence-level text classification datasets (AG's News, Movie Review), we consider multiple types of victim models to attack, including bi-directional word LSTM, fine-tuned BERT-base, BERT-large, XLNet-base, and XLNet-large (Hochreiter & Schmidhuber, 1997; Devlin et al., 2019; Yang et al., 2019). For the document-level text classification datasets, we focus on fine-tuned BERT-base, BERT-large, and XLNet-large as the victim model. For the textual entailment datasets, we focus on fine-tuned BERT-base as the victim model. We fine-tune BERT-large and XLNet-large models following the training protocol in TextAttack API (refer to our Github repository). For other models, we utilize publicly released models from Hugging Face and TextAttack Framework (Wolf et al., 2020; Morris et al., 2020), following the practice in Maheshwary et al. (2021). The average text length and the original classification accuracy of the victim models we consider for each dataset are summarized in Table 13.

For the protein classification dataset, we train three ASGD Weight-Dropped LSTM (AWD-LSTM) models (Merity et al., 2018), each corresponding to the classification task for one of the three levels, following the training protocol of Strodthoff et al. (2020). Then, we consider these AWD-LSTM models as victim models. We note that the average length of the test samples selected from EC50 is 411.3 and the original classification accuracy of the victim models for levels 0, 1, and 2 are 96.0%, 93.2%, and 92.4%, respectively.

Table 13: The average text length and the classification accuracy of the victim models for each NLP dataset.

Dataset	Model	Avg. Len.	Acc. (%)
AG	BERT-base	39.0	93.8
	LSTM		90.4
MR	XLNet-large	18.2	88.8
	BERT-large		84.8
	XLNet-base		87.2
	BERT-base		83.4
	LSTM		78.6
IMDB	XLNet-large	242.5	96.0
	BERT-large		93.8
	BERT-base		91.4
Yelp	XLNet-large	136.0	98.6
	BERT-large		98.6
	BERT-base		97.8
MNLI	BERT-base	29.7	87.4
QNLI	BERT-base	37.6	92.2

C.5. Details about Comparison in NLP

Table 14: Attack results of the original TextFooler (TF-orig) and TextFooler-fixed (TF-fixed) on the setence-classification datasets.

Model	Dataset	Method	ASR (%)	MR (%)	Qrs
BERT-base	AG	TF-orig	84.2	24.6	342
		TF-fixed	84.7	24.9	346
	MR	TF-orig	88.7	20.0	115
		TF-fixed	89.2	20.0	115
LSTM	AG	TF-orig	94.7	17.4	229
		TF-fixed	94.9	17.3	228
	MR	TF-orig	98.2	13.6	72
		TF-fixed	98.2	13.6	72
XLNet	MR	TF-orig	95.2	18.1	101
		TF-fixed	95.0	18.0	101

Table 15: Attack results of the original TextFooler (TF-orig) and TextFooler-fixed (TF-fixed) on the document-classification datasets against BERT-base models.

Model	Method	ASR (%)	MR (%)	Qrs
IMDB	TF-orig	98.9	8.5	706
	TF-fixed	99.1	8.6	712
Yelp	TF-orig	92.8	10.8	460
	TF-fixed	93.5	11.1	461

Here, we explain how we achieve a fair comparison with the baseline methods, focusing on the word substitution methods and attack spaces.

C.5.1. Comparison with PWWS and PSO

PWWS utilizes the word substitution method based on WordNet, which is a lexical database for the English language containing synonym sets for English words. Note that the WordNet-Based synonym set for a word w is built without considering the context of the word in the text. Thus, we can write the synonym set by $\mathcal{C}^{\text{wordnet}}(w)$. For a fair comparison, we compare BBA and PWWS with the same word synonym sets. Specifically, we apply BBA on the attack space defined by the product space of WordNet-based synonym sets, $\prod_{i=0}^{l-1} \mathcal{C}^{\text{wordnet}}(w_i)$.

PSO proposes a word-substitution model based on the semantic labels of words, also referred to as sememes, and applies particle swarm optimization on the product space of sememe sets corresponding to the original words. For a fair comparison, we apply BBA on the same attack space with PSO.

Table 16: Attack results against XLNet-large and BERT-large on IMDB, Yelp, and MR.

(a) WordNet

Dataset	Model	Method	ASR (%)	MR (%)	Qrs
IMDB	XLNet-large	PWWS	98.8	5.7	1697
		BBA	**99.8**	**5.4**	**573**
	BERT-large	PWWS	98.1	4.9	1661
		BBA	**100.0**	**4.6**	**619**
Yelp	XLNet-large	PWWS	94.5	10.8	1107
		BBA	**98.2**	**9.4**	**485**
	BERT-large	PWWS	75.1	**7.7**	1206
		BBA	**94.7**	**7.7**	**657**
MR	XLNet-large	PWWS	82.7	14.9	145
		BBA	**87.6**	**14.8**	**98**
	BERT-large	PWWS	80.0	14.6	146
		BBA	**85.1**	**14.2**	**110**

(b) Embedding

Dataset	Model	Method	ASR (%)	MR (%)	Qrs
IMDB	XLNet-large	TF	99.0	8.8	789
		BBA	**99.6**	**6.9**	**638**
	BERT-large	TF	97.2	7.0	660
		BBA	**98.7**	**6.4**	**559**
Yelp	XLNet-large	TF	95.9	16.7	635
		BBA	**99.4**	**12.6**	**475**
	BERT-large	TF	68.8	14.1	924
		BBA	**93.9**	**11.3**	**548**
MR	XLNet-large	TF	94.6	18.4	103
		BBA	**96.9**	**16.2**	**68**
	BERT-large	TF	90.8	18.7	111
		BBA	**94.8**	**17.0**	**71**

C.5.2. Comparison with TextFooler

TextFooler proposes an embedding-based synonym set that considers both word and sentence similarities. Thus, we can express the synonym set of a word w' as a function of the word and the sentence s' containing the word, denoted by $\mathcal{C}^{\text{Emb}}(w', s')$. Note that the synonym set can change dynamically during the optimization.

We first suggest TextFooler-fixed, a variant of the TextFooler method whose synonym set is defined by $\mathcal{C}^{\text{EmbFix}}(w') \triangleq \mathcal{C}^{\text{Emb}}(w', s)$, where s is the original input sentence. As defined above, the synonym set of TextFooler-fixed is fixed during the optimization. Tables 14 and 15 shows that TextFooler-fixed achieves attack performance comparable to the original TextFooler method. For a fair comparison, we apply BBA on the product space $\prod_{i=0}^{l-1} \mathcal{C}^{\text{EmbFix}}(w_i)$ and compare BBA with the TextFooler-fixed method instead of the original TextFooler method to use the same synonym sets for the attack.

C.5.3. Comparison with LSH

We compare BBA with the LSH method using each of all three word substitution methods, WordNet, Embedding, and HowNet. For experiments using the Embedding-based word substitution method, we set the attack space to the same as the TextFooler-fixed method.

D. Additional Results

D.1. Additional Baselines and Target Models

Table 16 shows that BBA outperforms PWWS and TextFooler in Movie Review, Yelp, and IMDB datasets against BERT-large and XLNet-large models. Also, BBA achieve state-of-the-art performance in all three evaluation metrics on textual entailment tasks, as shown in Table 17.

Table 18 shows that BBA outperforms Garg & Ramakrishnan (2020) (BAE) in various datasets against BERT-base.

Table 17: Attack results for BERT-base models on textual entailment datasets. $\mathcal{C}$ denotes the word substitution method used in attack.

$\mathcal{C}$	Dataset	Method	ASR (%)	MR (%)	Qrs
WordNet	MNLI	PWWS	84.4	6.1	102
		BBA	**85.1**	**5.9**	**42**
	QNLI	PWWS	67.7	9.3	211
		BBA	**73.3**	**8.8**	**168**
Embedding	MNLI	TF	92.5	7.3	77
		BBA	**93.8**	**6.2**	**49**
	QNLI	TF	83.7	11.1	172
		BBA	**87.0**	**9.4**	**157**
HowNet	MNLI	PSO	**85.6**	6.2	951
		BBA	**85.6**	**5.6**	**100**
	QNLI	PSO	68.6	13.8	31642
		BBA	**70.1**	**9.2**	**2369**

D.2. The Effect of Varying Block Size w/, w/o Post-Optimization

We investigate the effect of varying the block size m on the evaluation metrics for the protein classification task, with and without the post-optimization step. Table 19 shows that BBA with a larger block size achieves a higher attack success rate with fewer queries but higher modification rate. The attack results with the post-optimization process show that the post-optimization successfully reduces the modification rate.

Table 18: Comparison with BAE against BERT-base model on various datasets.

Dataset	Method	ASR (%)	MR (%)	Qrs
MR	BAE	81.3	16.1	87
	BBA	**93.1**	**14.2**	**57**
AG	BAE	53.7	23.1	329
	BBA	**73.6**	**21.6**	**246**
Yelp	BAE	88.1	11.5	402
	BBA	**97.8**	**10.1**	**266**
IMDB	BAE	95.2	8.9	592
	BBA	**98.9**	**5.3**	**353**

Table 19: Attack results for a AWD-LSTM model on EC50 level 1 dataset while varying the block size m.

		w/o Post-optimization		w/ Post-optimization	
m	ASR (%)	MR (%)	Qrs	MR (%)	Qrs
2	94.6	3.5	186	1.7	428
5	97.6	4.6	137	1.8	315
10	99.6	6.2	107	2.1	272
20	99.8	9.3	105	2.3	293
40	99.6	15.0	86	2.5	326
80	100.0	25.1	57	2.6	403

D.3. Query Efficiency of BBA

We provide more cumulative distribution plots against BERT-base models on Movie Review, IMDB, and MNLI datasets. Figures 6 to 8 show that BBA can find the adversarial sequences using less number of queries than the baseline methods also on the Movie Review, IMDB, and MNLI datasets.

D.4. Qualitative Results

Attack examples of Movie Review, Yelp, IMDB, MNLI, and QNLI datasets in Tables 20 and 21 show that BBA successfully generates semantically consistent adversarial texts while baseline methods fail to attack or generate adversarial sequences with high modification rates.

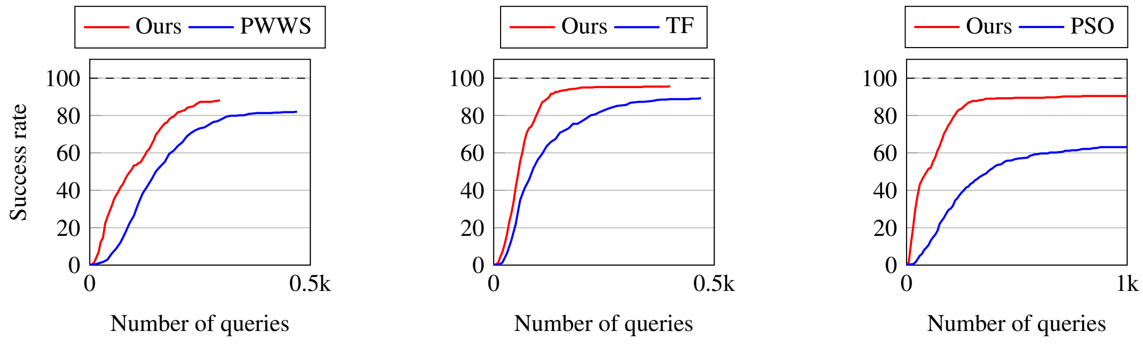

Figure 6: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the Movie Review dataset.

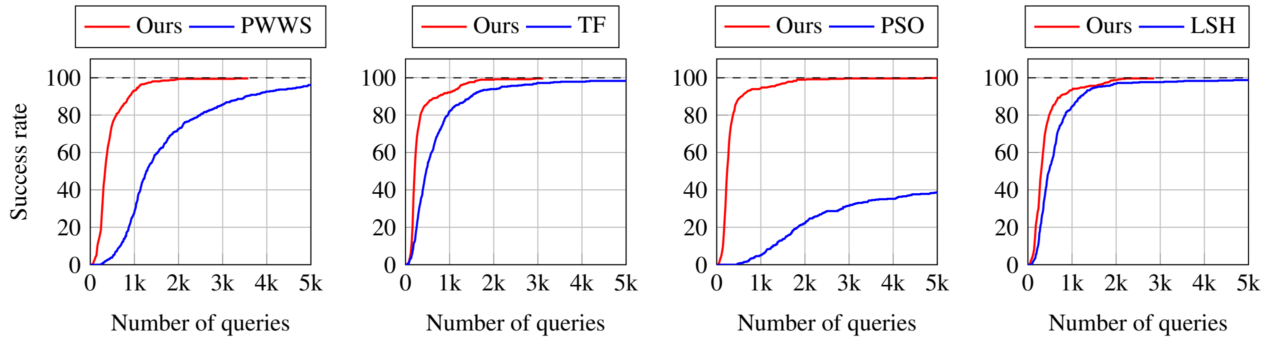

Figure 7: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the IMDB dataset. We use the HowNet based word substitution when comparing our method against LSH.

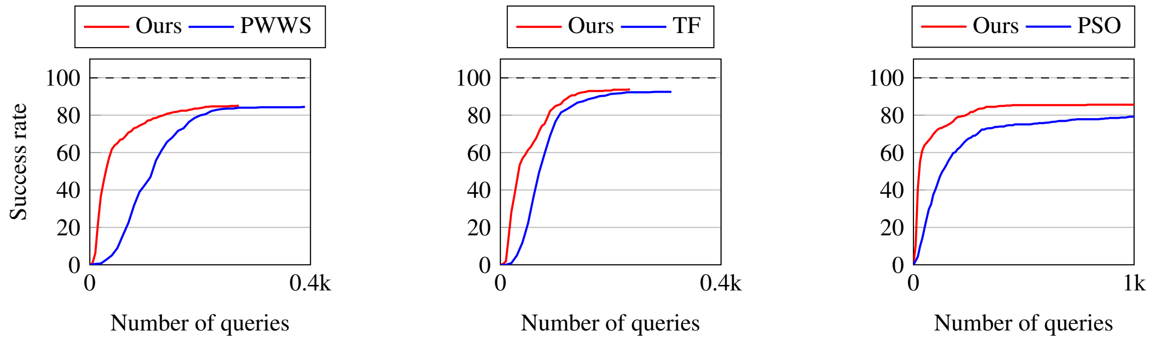

Figure 8: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the MNLI dataset.

Table 20: Examples of the original and their adversarial sequences from MR, Yelp, and IMDB against BERT-base models.

Sentence-Level Text Classification (Movie Review)		Label
Orig	suffers from a decided lack of creative storytelling.	Negative
BBA	*undergo* from a decided *dearth* of creative storytelling.	Positive
TF	-	Fail
Document-Level Text Classification (Yelp)		**Label**
Orig	I had never been here before, but I'm glad I tried it out. It was the best massage that I've ever had.	Positive
BBA	I had never been here before, but I'm glad I tried it out. It was the *allright massaging* that I've ever had.	Negative
TF	I had never been here before, but I'm *excited me* tried it out. *His* was the *alright* massage that I've ever *received*.	Negative
Orig	Not sure if they are closed for business but none of my calls were returned when I left a few VM's. Will update this if they ever do call back.	Negative
BBA	Not sure if they are closed for business but none of my *ask* were returned when I left a few VM's. Will *refreshing* this if they ever do call back.	Positive
TF	Not sure if they are closed for *companies* but none of my *appeals* were returned when I *going* a *short* VM's. *Dedication* update this if they ever do call back.	Positive
Document-Level Text Classification (IMDB)		**Label**
Orig	I rented this movie, because I noticed the cover in the video rental store. I saw Nolte, Connely, Madsen, 40's time setting, and thought "hmm, can't be too bad." Unfortunately, after watching it, my impression was "not too good". Its kind of a Chinatown ripoff, but the worst part is that other than Nolte, the other members of the squad didn't get enough screen time. But its a decent movie to see once I guess. And Melanie's role was small enough that she wasn't given a chance to be a nuisance.	Negative
BBA	I rented this movie, because I noticed the cover in the video rental store. I saw Nolte, Connely, Madsen, 40's time setting, and thought "hmm, can't be too bad." *Unluckily*, after watching it, my impression was "not too good". Its kind of a Chinatown ripoff, but the worst part is that other than Nolte, the other members of the squad didn't get enough screen time. But its a *honest* movie to see once I guess. And Melanie's role was small enough that she wasn't given a chance to be a nuisance.	Positive
PSO	I rented this *documentary*, because I noticed the cover in the video rental store. I *snapped* Nolte, Connely, Madsen, 40's time setting, and thought "hmm, can't be too bad." *Unluckily*, after watching it, my *career* was "not too good". Its kind of a Chinatown ripoff, but the *seediest farewell* is that other than Nolte, the other members of the *world* didn't get *substantial* screen time. But its a *honest ballad* to see once I guess. And Melanie's role was *humble* enough that she wasn't given a *opportunity* to be a *headache*.	Positive

Table 21: Examples of the original and their adversarial sequences from MNLI and QNLI against BERT-base models.

Textual Entailment (MNLI)		Label
Premise	that's really true a lot of it is um the color certain colors seem to be more acceptable.	
Orig	It doesn't make a difference what color one wears.	Neutral
BBA	It doesn't make a *dispute* what color one wears.	Entailment
TF	*His* doesn't *do* a difference what *colourful* one *focuses*.	Entailment
Premise	I turned a curve and I was just in time to see him ring the bell and get admitted to the house.	
Orig	I turned a curve and was just in time to see him ringing the big brass bell, echoing as he was admitted to the house.	Entailment
BBA	I turned a curve and was just in time to see him ringing the big brass bell, *invoking* as he was admitted to the *hostels*.	Neutral
TF	I turned a curve and was just in time to see him *cyclic* the big brass *chime*, *noting* as he was *hospitalised* to the *sarcophagus*.	Neutral
Textual Entailment (QNLI)		**Label**
Question	What was the main idea of James Hutton's paper?	
Orig	In his paper, he explained his theory that the Earth must be much older than had previously been supposed in order to allow enough time for mountains to be eroded and for sediments to form new rocks at the bottom of the sea, which in turn were raised up to become dry land.	Entailment
BBA	In his *journals*, he explained his theory that the Earth must be much older than had previously been supposed in order to allow enough time for mountains to be eroded and for sediments to form new rocks at the bottom of the sea, which in turn were raised up to become dry land.	Not Entailment
TF	In his paper, he explained his theory that the Earth must be much older than had previously been supposed in *writs* to *activation* enough *timing* for mountains to be eroded and for sediments to form new rocks at the bottom of the sea, which in turn were raised up to become dry land.	Not Entailment