



AFRL-AFOSR-JP-TR-2023-0050

Classification Attack Detection via Class-specific Visual Context Optimization

Hyun Oh Song
SEOUL NATIONAL UNIVERSITY
599 GWANAK-RO, GWANAK-GU
SEOUL, , 151742
KOR

12/21/2022
Final Technical Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
Air Force Office of Scientific Research
Asian Office of Aerospace Research and Development
Unit 45002, APO AP 96338-5002

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE 20221221		2. REPORT TYPE Final		3. DATES COVERED <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none;">START DATE 20200925</td> <td style="width: 50%; border: none;">END DATE 20220924</td> </tr> </table>		START DATE 20200925	END DATE 20220924
START DATE 20200925	END DATE 20220924						
4. TITLE AND SUBTITLE Classification Attack Detection via Class-specific Visual Context Optimization							
5a. CONTRACT NUMBER		5b. GRANT NUMBER FA2386-20-1-4043		5c. PROGRAM ELEMENT NUMBER			
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER			
6. AUTHOR(S) Hyun Oh Song							
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SEOUL NATIONAL UNIVERSITY 599 GWANAK-RO, GWANAK-GU SEOUL 151742 KOR				8. PERFORMING ORGANIZATION REPORT NUMBER			
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AOARD UNIT 45002 APO AP 96338-5002			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/AFOSR IOA		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-AFOSR-JP-TR-2023-0050		
12. DISTRIBUTION/AVAILABILITY STATEMENT A Distribution Unlimited: PB Public Release							
13. SUPPLEMENTARY NOTES							
14. ABSTRACT This final performance report describes three branches of research results performed under the AFOSR award FA2386-20-1-4043. We made significant progress towards state of the art adversarial defense framework for man-in-the-middle (MitM) adversarial attack on neural network classifiers, fast and accurate black-box adversarial attack algorithm for discrete sequential data (e.g. natural languages texts, protein sequences, etc.), and robust policy optimization algorithm for offline reinforcement learning. The research results were disseminated through (i) publications at top publication venues in AI (NeurIPS, AAAI, ICML), (ii) open sourced codes on Github for reproducibility and dissemination, and (iii) lectures for undergraduate and graduate level AI courses at Seoul National University.							
15. SUBJECT TERMS							
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR		18. NUMBER OF PAGES 55		
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U					
19a. NAME OF RESPONSIBLE PERSON AKIRA NAMATAME				19b. PHONE NUMBER (Include area code) 3152277010			



AFOSR Final Performance Report

Award Number: FA2386-20-1-4043
Principle Investigator: Hyun Oh Song
Affiliation: Seoul National University

December 20, 2022

1 Abstract

This final performance report describes three branches of research results performed under the AFOSR award FA2386-20-1-4043. We made significant progress towards state of the art adversarial defense framework for man-in-the-middle (MitM) adversarial attack on neural network classifiers, fast and accurate black-box adversarial attack algorithm for discrete sequential data (e.g. natural languages texts, protein sequences, etc.), and robust policy optimization algorithm for offline reinforcement learning. The research results were disseminated through (i) publications at top publication venues in AI (NeurIPS, AAAI, ICML), (ii) open sourced codes on Github for reproducibility and dissemination, and (iii) lectures for undergraduate and graduate level AI courses at Seoul National University.

2 Accomplishments

We made significant progress in (1) robust algorithms for offline reinforcement learning [1], (2) defense against adversarial attacks on image classifiers [14], and (3) novel and query efficient algorithms for adversarial attacks on discrete sequential data [11].

We published our findings at top tier publication venues in artificial intelligence including NeurIPS 2021 [1], AAAI 2022 [14], and ICML 2022 [11]. We also made the source code publicly available on Github. Table 1 summarizes the accomplishments.

Table 1: Summary of accomplishments. We acknowledged AFOSR funding for all three publications.

Research	Publication venue	Publication link	Code link
Offline RL	NeurIPS 2021	paper	https://github.com/snu-mllab/EDAC
Adversarial defense	AAAI 2022	paper	https://github.com/snu-mllab/preemptive-robustification
Discrete adversarial attack	ICML 2022	paper	https://github.com/snu-mllab/DiscreteBlockBayesAttack

In the following subsections, we summarize the objectives and the sketch of the accomplishments and results for each research. Please refer to the attached respective full papers for complete details and results.

2.1 Robust offline reinforcement learning (NeurIPS 2021)

Conventional reinforcement learning algorithms require interaction with the environment and learns control policies by trial and error. This can be prohibitive in real world applications such as autonomous driving and healthcare, as exploratory actions can lead to critical damage to the agent or the environment.

To this end, *offline reinforcement learning* (ORL) have been recently proposed [12]. In ORL, the goal is to learn policies given a large static dataset of previously collected interactions via human demonstrators or controlled policies and without further interactions with the environment. The core challenge in ORL arises when the coverage of the dataset is not sufficient and the neural network overestimates Q-values (expected return of a given state-action pair) of out-of-distribution (OOD) state-action pairs due to extrapolation. We proposed an uncertainty-based model-free ORL algorithm that effectively quantifies the uncertainty of the Q-value estimates of Q-function networks significantly mitigating the overestimation problem in ORL.

Our experiments on benchmark D4RL datasets showed the state of the art performance compared to existing ORL methods [4]. We experimented on D4RL MuJoCo Gym tasks for continuous robotic control (halfcheetah, hopper, walker2d) and D4RL Adroit tasks for 24-DoF robotic hand control.

2.2 Defense against adversarial attacks (AAAI 2022)

Neural networks exhibit vulnerability to adversarial examples [6], minute input perturbations intentionally crafted to mislead the networks to yield incorrect predictions. In this work, we consider adversarial defense framework for a man-in-the-middle (MitM) adversary that can intercept and add adversarial perturbations to the images uploaded during transmission.

We propose a novel bi-level optimization algorithm for finding robust points in the vicinity of natural images that are resistant to adversarial perturbations. The optimization criteria is to find a robust point x_r in the vicinity of the input image x_o which has the least drop in the classification loss when x_r is adversarially perturbed with a given budget ϵ . The proposed framework is to pre-emptively transform the input image x_o to optimized robust x_r before uploading it to the web online.

Our experiment on image classification benchmark datasets (CIFAR-10 and ImageNet [9, 10]) demonstrate that our framework can successfully robustify input images against a wide range of adversarial attacks (PGD, AutoAttack [3]) in both black-box and white-box settings.

2.3 Adversarial attacks on discrete sequential data (ICML 2022)

While a large body of works on blackbox adversarial attacks focus mainly on image domain with continuous attack spaces, developing adversarial attacks on discrete input spaces has been much less studied [7, 2]. This is mainly because developing a query-efficient black-box attack on discrete sequential data is quite challenging due to discrete nature of the attack spaces.

To this end, we propose a blockwise Bayesian attack framework, a query-efficient black-box attack based on *Bayesian Optimization*. We devise a novel block attack decomposition and query history subsampling techniques that successfully improve the query and computation efficiency without compromising the attack success rate.

Our experiments on text classification, textual entailment, and protein classification against various victim models demonstrate state of the art attack performance in comparison to recent baseline text attack methods (PWWS [15], TextFooler [8], LSH [13], BAE [5], PSO [16]).

3 Impacts

The award supported three graduate students in SNU MLLAB and helped them become independent researchers in machine learning. Graduate student Gaon An is the first author on offline reinforcement learning published at NeurIPS 2021 [1]. Graduate student Seungyong Moon is the first author on adversarial defense published at AAAI 2022 [14]. Graduate student Deokjae Lee is the first author on adversarial attacks on discrete sequence data published at ICML 2022 [11].

Furthermore, PI Hyun Oh Song teaches “Introduction to deep learning” course (M2177.0043) taught for senior undergraduate students and “Machine learning” course (4190.666) for graduate students at Seoul National University. The courses attract many undergraduate students from diverse departments including all fields in engineering, Mathematics, Statistics, and economics. The core methodologies from the three publications (all acknowledged AFOSR funding) were introduced as lecture materials during the course. This led several undergraduate students to pursue the research career as graduate students by joining our SNU MLLAB.

4 Technical Updates

The published papers in NeurIPS 2021 [1], AAAI 2022 [14], and ICML 2022 [11] are attached in the subsequent pages.

5 References

- [1] G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *NeurIPS*, 2021.

- [2] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pages 484–501. Springer, 2020.
- [3] F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.
- [4] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.
- [5] S. Garg and G. Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *EMNLP*, 2020.
- [6] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [7] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.
- [8] 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. In *AAAI*, 2020.
- [9] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [11] D. Lee, S. Moon, J. Lee, and H. O. Song. Query-efficient and scalable black-box adversarial attacks on discrete sequential data via bayesian optimization. In *International Conference on Machine Learning (ICML)*, pages 12478–12497. PMLR, 2022.
- [12] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- [13] R. Maheshwary, S. Maheshwary, and V. Pudi. A strong baseline for query efficient attacks in a black box setting. In *EMNLP*, 2021.
- [14] S. Moon, G. An, and H. O. Song. Preemptive image robustification for protecting users against man-in-the-middle adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.
- [15] S. Ren, Y. Deng, K. He, and W. Che. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, 2019.
- [16] Y. Zang, C. Yang, F. Qi, Z. Liu, M. Zhang, Q. Liu, and M. Sun. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, 2020.

Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble

Gaon An^{*12}, Seungyong Moon^{*12}, Jang-Hyun Kim¹², Hyun Oh Song^{†123}

Seoul National University¹

Neural Processing Research Center²

DeepMetrics³

{white0234, symoon11, janghyun, hyunoh}@mllab.snu.ac.kr

Abstract

Offline reinforcement learning (offline RL), which aims to find an optimal policy from a previously collected static dataset, bears algorithmic difficulties due to function approximation errors from out-of-distribution (OOD) data points. To this end, offline RL algorithms adopt either a constraint or a penalty term that explicitly guides the policy to stay close to the given dataset. However, prior methods typically require accurate estimation of the behavior policy or sampling from OOD data points, which themselves can be a non-trivial problem. Moreover, these methods under-utilize the generalization ability of deep neural networks and often fall into suboptimal solutions too close to the given dataset. In this work, we propose an uncertainty-based offline RL method that takes into account the confidence of the Q-value prediction and does not require any estimation or sampling of the data distribution. We show that the clipped Q-learning, a technique widely used in online RL, can be leveraged to successfully penalize OOD data points with high prediction uncertainties. Surprisingly, we find that it is possible to substantially outperform existing offline RL methods on various tasks by simply increasing the number of Q-networks along with the clipped Q-learning. Based on this observation, we propose an ensemble-diversified actor-critic algorithm that reduces the number of required ensemble networks down to a tenth compared to the naive ensemble while achieving state-of-the-art performance on most of the D4RL benchmarks considered.

1 Introduction

Over the recent years, deep reinforcement learning (deep RL) has achieved considerable success in various domains such as robotics [20], recommendation systems [6], and strategy games [26]. However, a major drawback of RL algorithms is that they adopt an active learning procedure, where training steps require active interactions with the environment. This trial-and-error procedure can be prohibitive when scaling RL to real-world applications such as autonomous driving and healthcare, as exploratory actions can cause critical damage to the agent or the environment [19]. *Offline* RL, also known as *batch* RL, aims to overcome this problem by learning policies using only previously collected data without further interactions with the environment [2, 11, 19].

Even though offline RL is a promising direction to lead a more *data-driven* way of solving RL problems, recent works show offline RL faces new algorithmic challenges [19]. Typically, if the coverage of the dataset is not sufficient, vanilla RL algorithms suffer severely from extrapolation

^{*}First two authors have equal contributions

[†]Corresponding author

error, overestimating the Q-values of out-of-distribution (OOD) state-action pairs [15]. To this end, most offline RL methods apply some constraints or penalty terms on top of the existing RL algorithms to enforce the learning process to be more conservative. For example, some prior works explicitly regularize the policy to be close to the behavior policy that was used to collect the data [11, 15]. A more recent work instead penalizes the Q-values of OOD state-action pairs to enforce the Q-values to be more pessimistic [16].

While these methods achieve significant performance gains over vanilla RL methods, they either require an estimation of the behavior policy or explicit sampling from OOD data points, which themselves can be non-trivial to solve. Furthermore, these methods do not utilize the generalization ability of the Q-function networks and prohibit the agent from approaching any OOD state-actions without any consideration on whether they are good or bad. However, if we can identify OOD data points where we can predict their Q-values with high confidence, it is more effective not to restrain the agent from choosing those data points.

From this intuition, we propose an uncertainty-based model-free offline RL method that effectively quantifies the uncertainty of the Q-value estimates by an ensemble of Q-function networks and does not require any estimation or sampling of the data distribution. To achieve this, we first show that a well-known technique from online RL, the clipped Q-learning [10], can be successfully leveraged as an uncertainty-based penalization term. Our experiments reveal that we can achieve state-of-the-art performance on various offline RL tasks by solely using this technique with increased ensemble size. To further improve the practical usability of the method, we develop an ensemble diversifying objective that significantly reduces the number of required ensemble networks. We evaluate our proposed method on D4RL benchmarks [9] and verify that the proposed method outperforms the previous state-of-the-art by a large margin on various types of environments and datasets.

2 Preliminaries

We consider an environment formulated as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, T, r, d_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s' | s, a)$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, d_0 is the initial state distribution, and $\gamma \in (0, 1]$ is the discount factor. The goal of reinforcement learning is to find an optimal policy $\pi(a | s)$ that maximizes the cumulative discounted reward $\mathbb{E}_{s_0 \sim d_0(\cdot), a_t \sim \pi(\cdot | s_t)} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where $s_0 \sim d_0(\cdot)$, $a_t \sim \pi(\cdot | s_t)$, and $s_{t+1} \sim T(\cdot | s_t, a_t)$.

One of the major approaches for obtaining such a policy is Q-learning [12, 20] which learns a state-action value function $Q_\phi(s, a)$ parameterized by a neural network that represents the expected cumulative discounted reward when starting from state s and action a . Standard actor-critic approach [14] learns this Q-function by minimizing the Bellman residual $(Q_\phi(s, a) - \mathcal{B}^{\pi_\theta} Q_\phi(s, a))^2$, where $\mathcal{B}^{\pi_\theta} Q_\phi(s, a) = \mathbb{E}_{s' \sim T(\cdot | s, a)} [r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_\theta(\cdot | s')} Q_\phi(s', a')]$ is the Bellman operator. In the context of offline RL, where transitions are sampled from a static dataset $\mathcal{D}$, the objective for the Q-network becomes minimizing

$$J_q(Q_\phi) := \mathbb{E}_{(s, a, s') \sim \mathcal{D}} \left[\left(Q_\phi(s, a) - (r(s, a) + \gamma \mathbb{E}_{a' \sim \pi_\theta(\cdot | s')} [Q_{\phi'}(s', a')]) \right)^2 \right], \quad (1)$$

where $Q_{\phi'}$ represents the target Q-network softly updated for algorithmic stability [20]. The policy, which is also parameterized by a neural network, is updated in an alternating fashion to maximize the expected Q-value: $J_p(\pi_\theta) := \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\theta(\cdot | s)} [Q_\phi(s, a)]$.

However, as the policy is updated to maximize the Q-values, the actions a' sampled from the current policy in Equation (1) can be biased towards OOD actions with erroneously high Q-values. In the offline RL setting, such errors cannot be corrected by feedback from the environment as in online RL. To handle the error propagation from these OOD actions, most offline RL algorithms regularize either the policy [11, 15] or the Q-function [16] to be biased towards the given dataset. However, the policy regularization methods typically require an accurate estimation of the behavior policy. The previous state-of-the-art method CQL [16] instead learns conservative Q-values without estimating the behavior policy by penalizing the Q-values of OOD actions by

$$\min_{\phi} J_q(Q_\phi) + \alpha \left(\mathbb{E}_{s \sim \mathcal{D}, a \sim \mu(\cdot | s)} [Q_\phi(s, a)] - \mathbb{E}_{(s, a) \sim \mathcal{D}} [Q_\phi(s, a)] \right),$$

where μ is an approximation of the policy that maximizes the current Q-function. While CQL does not need explicit behavior policy estimation, it requires sampling from an appropriate action distribution $\mu(\cdot | \mathbf{s})$.

3 Uncertainty penalization with Q-ensemble

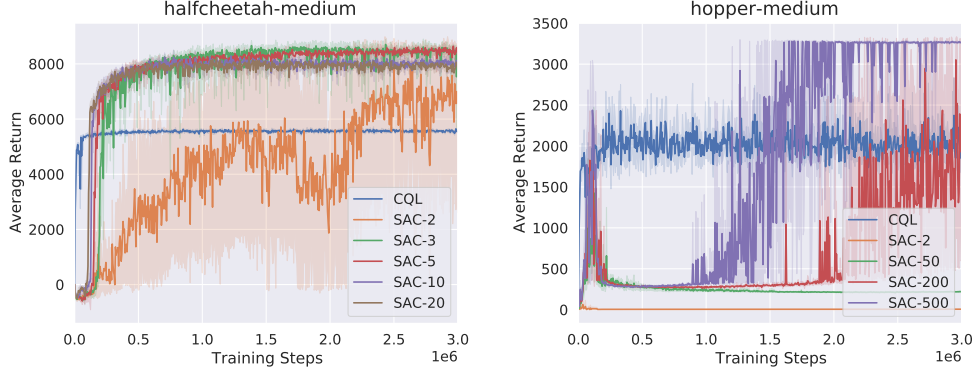


Figure 1: Performance of SAC- N on halfcheetah-medium and hopper-medium datasets while varying N , compared to CQL. ‘Average Return’ denotes the undiscounted return of each policies on evaluation. Results averaged over 4 seeds.

In this section, we turn our attention to a conventional technique from online RL, Clipped Double Q-learning [10], which uses the minimum value of two parallel Q-networks as the Bellman target: $y = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\cdot | \mathbf{s}')} [\min_{j=1,2} Q_{\phi'_j}(\mathbf{s}', \mathbf{a}')]]$. Although this technique was originally proposed in online RL to mitigate the overestimation from general prediction errors, some offline RL algorithms [11, 15, 28] also utilize this technique to enforce their Q-value estimates to be more pessimistic. However, the isolated effect of the clipped Q-learning in offline RL was not fully analyzed in the previous works, as they use the technique only as an auxiliary term that adds up to their core methods.

To examine the ability of clipped Q-learning to prevent the overestimation in offline RL on its own, we modify SAC [12] by increasing the number of Q-ensembles from 2 to N :

$$\begin{aligned} \min_{\phi_i} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}} \left[\left(Q_{\phi_i}(\mathbf{s}, \mathbf{a}) - \left(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\cdot | \mathbf{s}')} \left[\min_{j=1, \dots, N} Q_{\phi'_j}(\mathbf{s}', \mathbf{a}') - \beta \log \pi_\theta(\mathbf{a}' | \mathbf{s}') \right] \right) \right)^2 \right] \\ \max_{\theta} \mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi_\theta(\cdot | \mathbf{s})} \left[\min_{j=1, \dots, N} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) - \beta \log \pi_\theta(\mathbf{a} | \mathbf{s}) \right], \end{aligned} \quad (2)$$

for $i = 1, \dots, N$. We denote this modified algorithm as SAC- N .

Figure 1 shows the preliminary experiments on D4RL halfcheetah-medium and hopper-medium datasets [9] while varying N . Note that these datasets are constructed from suboptimal behavior policies. Surprisingly, as we gradually increase N , we can successfully find policies that outperform the previous state-of-the-art method (CQL) by a large margin. In fact, as we will present in Section 5, SAC- N outperforms CQL on various types of environments and data-collection policies.

To understand why this simple technique works so well, we can first interpret the clipping procedure (choosing the minimum value from the ensemble) as penalizing state-action pairs with high-variance Q-value estimates, which encourages the policy to favor actions that appeared in the dataset [11]. The dataset samples will naturally have lower variance compared to the OOD samples as the Bellman residual term in Equation (2) explicitly aligns the Q-value predictions for the dataset samples. More formally, we can regard this difference in variance as accounting for *epistemic uncertainty* [8] which refers to the uncertainty stemming from limited data and knowledge.

Utilization of the clipped Q-value relates to methods that consider the confidence bound of the Q-value estimates [24]. Online RL methods typically utilize the Q-ensemble to form an optimistic estimate of the Q-value, by adding the standard deviation to the mean of the Q-ensembles [18]. This optimistic Q-value, also known as the upper-confidence bound (UCB), can encourage the exploration

of unseen actions with high uncertainty. However, in offline RL, the dataset available during training is fixed, and we have to focus on *exploiting* the given data. For this purpose, it is natural to utilize the lower-confidence bound (LCB) of the Q-value estimates, for example by subtracting the standard deviation from the mean, which allows us to avoid risky state-actions.

The clipped Q-learning algorithm, which chooses the worst-case Q-value instead to compute the pessimistic estimate, can also be interpreted as utilizing the LCB of the Q-value predictions. Suppose $Q(\mathbf{s}, \mathbf{a})$ follows a Gaussian distribution with mean $m(\mathbf{s}, \mathbf{a})$ and standard deviation $\sigma(\mathbf{s}, \mathbf{a})$. Also, let $\{Q_j(\mathbf{s}, \mathbf{a})\}_{j=1}^N$ be realizations of $Q(\mathbf{s}, \mathbf{a})$. Then, we can approximate the expected minimum of the realizations following the work of Royston [23] as

$$\mathbb{E} \left[\min_{j=1, \dots, N} Q_j(\mathbf{s}, \mathbf{a}) \right] \approx m(\mathbf{s}, \mathbf{a}) - \Phi^{-1} \left(\frac{N - \frac{\pi}{8}}{N - \frac{\pi}{4} + 1} \right) \sigma(\mathbf{s}, \mathbf{a}), \quad (3)$$

where Φ is the CDF of the standard Gaussian distribution. This relation indicates that using the clipped Q-value is similar to penalizing the ensemble mean of the Q-values with the standard deviation scaled by a coefficient dependent on N .

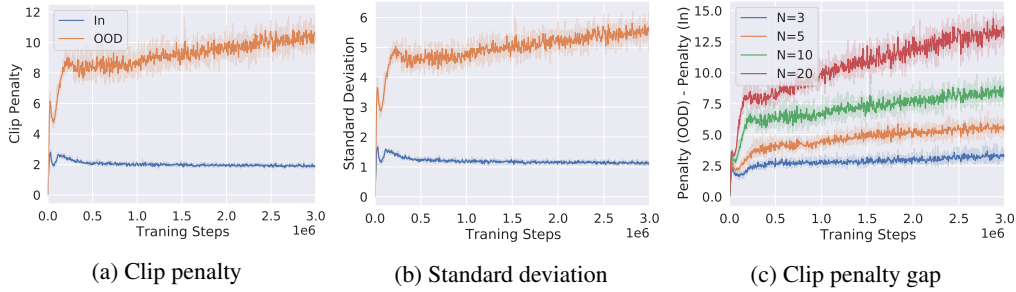


Figure 2: (a) and (b) each plots the size of the clip penalty and the standard deviation of the Q-value estimates for in-distribution (behavior) and OOD (random) actions while training SAC-10 on halfcheetah-medium dataset. (c) plots the gap of the clip penalty between the in-distribution and OOD actions while varying N . Results averaged over 4 seeds.

We now move on to the empirical analysis of the clipped Q-learning. Figure 2a compares the strength of the uncertainty penalty on in-distribution and OOD actions. Specifically, we compare actions sampled from two types of policies: (1) the behavior policy which was used to collect the dataset, and (2) the random policy which samples actions uniformly from the action space. For each policy, we measure the size of the penalty from the clipping as $\mathbb{E}_{\mathbf{s} \sim \mathcal{D}, \mathbf{a} \sim \pi(\cdot|\mathbf{s})} [\frac{1}{N} \sum_{j=1}^N Q_{\phi_j}(\mathbf{s}, \mathbf{a}) - \min_{j=1, \dots, N} Q_{\phi_j}(\mathbf{s}, \mathbf{a})]$. Figure 2a shows that the clipping term penalizes the random state-action pairs much stronger than the in-distribution pairs throughout the training. For comparison, we also measure the standard deviation of the Q-values for each policy. The results in Figure 2b show that as we conjectured, the Q-value predictions for the OOD actions have a higher variance. We also find that the size of the penalty and the standard deviation are highly correlated, as we noted in Equation (3).

As we observe that OOD actions have higher variance on Q-value estimates, the effect of increasing N becomes obvious: it strengthens the penalty applied to the OOD samples compared to the dataset samples. To verify this, we measured the relative penalty applied to the OOD samples in Figure 2c and found that indeed the OOD samples are penalized relatively further as N increases.

4 Ensemble gradient diversification

Even though SAC- N outperforms existing methods on various tasks, it sometimes requires an excessively large number of ensembles to learn stably (*e.g.*, $N = 500$ for hopper-medium). While investigating its reason, we found that the performance of SAC- N is negatively correlated with the degree to which the input gradients of Q-functions $\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$ are aligned, which increases with N . Figure 4 measures the minimum cosine similarity between the gradients of the Q-functions $\min_{i \neq j} \langle \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle$ to examine the alignment of the gradients while varying N on the D4RL hopper-medium dataset. The results imply that the performance of the learned policy degrades significantly when the Q-functions share a similar local structure.

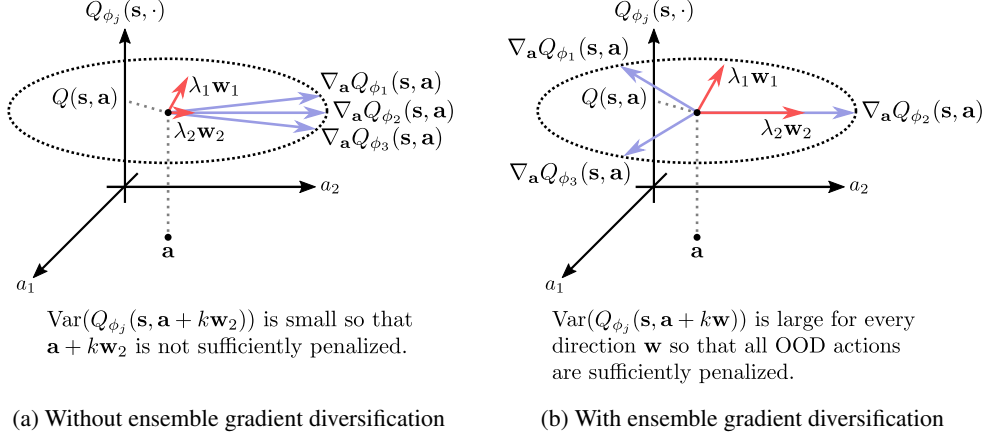


Figure 3: Illustration of the ensemble gradient diversification. The vector $\lambda_i \mathbf{w}_i$ represents the normalized eigenvector $\mathbf{w}_i$ of $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ multiplied by its eigenvalue λ_i .

We now show that the alignment of the input gradients can induce insufficient penalization of near-distribution data points, which leads to requiring a large number of ensemble networks. Let $\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$ be the gradient of the j -th Q-function with respect to the behavior action $\mathbf{a}$ and assume the gradient is normalized for simplicity. If the gradients of the Q-functions are well-aligned as illustrated in Figure 3a, then there exists a unit vector $\mathbf{w}$ such that the Q-values for the OOD actions along the direction of $\mathbf{w}$ have a low variance. To show this, we first assume the Q-value predictions for the in-distribution state-action pairs coincide, i.e., $Q_{\phi_j}(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$ for $j = 1, \dots, N$. Note that this can be optimized by minimizing the Bellman error. Then, using the first-order Taylor approximation, the sample variance of the Q-values at an OOD action along $\mathbf{w}$ can be represented as

$$\begin{aligned} \text{Var}(Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w})) &\approx \text{Var}(Q_{\phi_j}(\mathbf{s}, \mathbf{a}) + k \langle \mathbf{w}, \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle) \\ &= \text{Var}(Q(\mathbf{s}, \mathbf{a}) + k \langle \mathbf{w}, \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle) \\ &= k^2 \text{Var}(\langle \mathbf{w}, \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle) \\ &= k^2 \mathbf{w}^T \text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})) \mathbf{w}, \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes an inner-product, $k \in \mathbb{R}$, and $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ is the sample variance matrix for the input gradients $\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$. One interesting property of the variance matrix is that its total variance, which is equivalent to the sum of its eigenvalues, can be represented as a function of the norm of the average gradients by Lemma 1.

Lemma 1. The total variance of the matrix $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ is equal to $1 - \|\bar{\mathbf{q}}\|_2^2$, where $\bar{\mathbf{q}} = \frac{1}{N} \sum_{j=1}^N \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$.

Let $\lambda_{\min}$ be the smallest eigenvalue of $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ and $\mathbf{w}_{\min}$ be the corresponding normalized eigenvector. Also, let $\epsilon > 0$ be the value such that $\min_{i \neq j} \langle \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle = 1 - \epsilon$. Then, using Lemma 1, we can prove that the variance of the Q-values for an OOD action along $\mathbf{w}_{\min}$ is upper-bounded by some constant multiple of ϵ , which is given by Proposition 1.

Proposition 1. Suppose $Q_{\phi_j}(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$ and $Q_{\phi_j}(\mathbf{s}, \cdot)$ is locally linear in the neighborhood of $\mathbf{a}$ for all $j \in [N]$. Let $\lambda_{\min}$ and $\mathbf{w}_{\min}$ be the smallest eigenvalue and the corresponding normalized eigenvector of the matrix $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ and $\epsilon > 0$ be the value such that

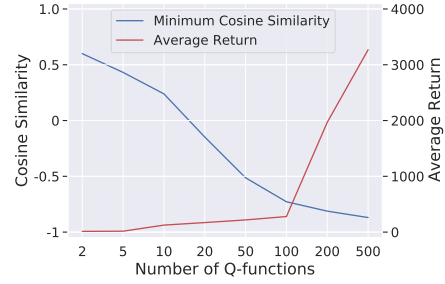


Figure 4: Plot of the minimum cosine similarity between the input gradients of Q-functions and the average return while varying the number of Q-functions.

$\min_{i \neq j} \langle \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle = 1 - \epsilon$. Then, the variance of the Q-values for an OOD action in the neighborhood along the direction of $\mathbf{w}_{\min}$ is upper-bounded as follows:

$$\text{Var} (Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w}_{\min})) \leq \frac{1}{|\mathcal{A}|} \frac{N-1}{N} k^2 \epsilon,$$

where $|\mathcal{A}|$ is the action space dimension.

We provide the proofs in Appendix A.1. Proposition 1 implies that if there exists such $\epsilon > 0$ that is small, which means the gradients of Q-function are well-aligned, then the variance of the Q-values for an OOD action along a specific direction will also be small. This in turn degrades the ability of the ensembles to penalize OOD actions, which ultimately leads to requiring a large number of ensemble networks.

To address this problem, we propose a regularizer that effectively increases the variance of the Q-values for near-distribution OOD actions. Note that the variance is lower-bounded by some constant multiple of the smallest eigenvalue $\lambda_{\min}$:

$$\begin{aligned} \text{Var} (Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w})) &\approx k^2 \mathbf{w}^\top \text{Var} (\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})) \mathbf{w} \\ &\geq k^2 \mathbf{w}_{\min}^\top \text{Var} (\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})) \mathbf{w}_{\min} \\ &= k^2 \lambda_{\min}. \end{aligned}$$

Therefore, an obvious way to increase this variance is to maximize the smallest eigenvalue of $\text{Var} (\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$, which can be formulated as

$$\underset{\phi}{\text{maximize}} \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} [\lambda_{\min} (\text{Var} (\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})))] ,$$

where ϕ denotes the collection of the parameters $\{\phi_j\}_{j=1}^N$. There are several methods to compute the smallest eigenvalue, such as the power method or the QR algorithm [27]. However, these iterative methods require constructing huge computation graphs, which makes optimizing the eigenvalue using back-propagation inefficient. Instead, we aim to maximize the sum of all eigenvalues, which is equal to the total variance. By Lemma 1, it is equivalent to minimizing the norm of the average gradients:

$$\underset{\phi}{\text{minimize}} \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[\left\langle \frac{1}{N} \sum_{i=1}^N \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \frac{1}{N} \sum_{j=1}^N \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \right\rangle \right]. \quad (4)$$

With simple modification, we can reformulate Equation (4) as diversifying the gradients of each Q-function network for in-distribution actions:

$$\underset{\phi}{\text{minimize}} J_{\text{ES}}(Q_{\phi}) := \mathbb{E}_{\mathbf{s}, \mathbf{a} \sim \mathcal{D}} \left[\frac{1}{N-1} \sum_{1 \leq i \neq j \leq N} \underbrace{\langle \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle}_{\text{ES}_{\phi_i, \phi_j}(\mathbf{s}, \mathbf{a})} \right].$$

Concretely, our final objective can be interpreted as measuring the pairwise alignment of the gradients using cosine similarity, which we denote as the Ensemble Similarity (ES) metric $\text{ES}_{\phi_i, \phi_j}(\mathbf{s}, \mathbf{a})$, and minimizing the ES values for every pair in the Q-ensemble with regard to the dataset state-actions. The illustration of the ensemble gradient diversification is shown in Figure 3b. Note that we instead maximize the total variance to reduce the computational burden. Nevertheless, the modified objective is closely related to maximizing the smallest eigenvalue. The detailed explanation can be found in Appendix A.2.

We name the resulting actor-critic algorithm as Ensemble-Diversified Actor Critic (EDAC) and present the detailed procedure in Algorithm 1 (differences with the original SAC algorithm marked in blue). Note that Algorithm 1 reduces to SAC- N when $\eta=0$, and further reduces to vanilla SAC when also $N=2$.

5 Experiments

We evaluate our proposed methods against the previous offline RL algorithms on the standard D4RL benchmark [9]. Concretely, we perform our evaluation on MuJoCo Gym (Section 5.1) and Adroit

Algorithm 1 Ensemble-Diversified Actor Critic (EDAC)

- 1: Initialize policy parameters θ , Q-function parameters $\{\phi_j\}_{j=1}^N$, target Q-function parameters $\{\phi'_j\}_{j=1}^N$, and offline data replay buffer $\mathcal{D}$
- 2: **repeat**
- 3: Sample a mini-batch $B = \{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')\}$ from $\mathcal{D}$
- 4: Compute target Q-values (shared by all Q-functions):

$$y(r, \mathbf{s}') = r + \gamma \left(\min_{j=1, \dots, N} Q_{\phi'_j}(\mathbf{s}', \mathbf{a}') - \beta \log \pi_{\theta}(\mathbf{a}' | \mathbf{s}') \right), \quad \mathbf{a}' \sim \pi_{\theta}(\cdot | \mathbf{s}')$$

- 5: Update each Q-function Q_{ϕ_i} with gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(\mathbf{s}, \mathbf{a}, r, \mathbf{s}') \in B} \left(\left(Q_{\phi_i}(\mathbf{s}, \mathbf{a}) - y(r, \mathbf{s}') \right)^2 + \frac{\eta}{N-1} \sum_{1 \leq i \neq j \leq N} \text{ES}_{\phi_i, \phi_j}(\mathbf{s}, \mathbf{a}) \right)$$

- 6: Update policy with gradient ascent using

$$\nabla_{\theta} \frac{1}{|B|} \sum_{\mathbf{s} \in B} \left(\min_{j=1, \dots, N} Q_{\phi_j}(\mathbf{s}, \tilde{\mathbf{a}}_{\theta}(\mathbf{s})) - \beta \log \pi_{\theta}(\tilde{\mathbf{a}}_{\theta}(\mathbf{s}) | \mathbf{s}) \right),$$

where $\tilde{\mathbf{a}}_{\theta}(\mathbf{s})$ is a sample from $\pi_{\theta}(\cdot | \mathbf{s})$ which is differentiable w.r.t. θ via the reparametrization trick.

- 7: Update target networks with $\phi'_i \leftarrow \rho \phi'_i + (1 - \rho) \phi_i$
-

(Section 5.2) domains. We consider the following baselines: SAC, the backbone algorithm of our method, CQL, the previous state-of-the-art on the D4RL benchmark, REM [2], an offline RL method which utilized Q-network ensemble on discrete control environments, and BC, the behavior cloning method. We evaluate each method under the normalized average return metric where the average return is scaled such that 0 and 100 each equals the performance of a random policy and an online expert policy. In addition to the performance evaluation, we compare the computational cost of each method (Section 5.3). For the implementation details of our algorithm and the baselines, please refer to Appendix B and Appendix C. Also, we provide more experiments such as comparison with more baselines, CQL with N Q-networks, and hyperparameter sensitivity from Appendix E to Appendix H.

5.1 Evaluation on D4RL MuJoCo Gym tasks

We first evaluate each method on D4RL MuJoCo Gym tasks which consist of three environments, halfcheetah, hopper, and walker2d, each with six datasets from different data-collecting policies. In detail, the considered policies are *random*: a uniform random policy, *expert*: a fully trained online expert, *medium*: a suboptimal policy with approximately 1/3 the performance of the expert, *medium-expert*: a mixture of medium and expert policies, *medium-replay*: the replay buffer of a policy trained up to the performance of the medium agent, and *full-replay*: the final replay buffer of the expert policy. Each dataset consists of 1M transitions except for medium-expert and medium-replay.

The experiment results in Table 1 show EDAC and SAC- N both outperform or are competitive with the previous state-of-the-art on all of the tasks considered. Notably, the performance gap is especially high for random, medium, and medium-replay datasets, where the performances of the previous works are relatively low. Both the proposed methods achieve average normalized scores over 80, reducing the gap with the online expert by 40% compared to CQL. While the performance of EDAC is marginally better than the performance of SAC- N , EDAC achieves this result with a much smaller Q-ensemble size. As noted in Figure 5, on hopper tasks, SAC- N requires 200 to 500 Q-networks, while EDAC requires less than 50.

Figure 6 compares the distance between the actions chosen by each method and the dataset actions. Concretely, we measure $\mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}, \hat{\mathbf{a}} \sim \pi_{\theta}(\cdot | \mathbf{s})} [\|\hat{\mathbf{a}} - \mathbf{a}\|_2^2]$ for EDAC, SAC- N , CQL, SAC-2, and a random policy on *-medium datasets. We find that our proposed methods choose from a more diverse range of actions compared to CQL. This shows the advantage of the uncertainty-based penalization which considers the prediction confidence other than penalizing all OOD actions.

Table 1: Normalized average returns on D4RL Gym tasks, averaged over 4 random seeds. CQL (Paper) denotes the results reported in the original paper.

Task Name	BC	SAC	REM	CQL (Paper)	CQL (Reproduced)	SAC-N (Ours)	EDAC (Ours)
halfcheetah-random	2.2±0.0	29.7±1.4	-0.8±1.1	35.4	31.3±3.5	28.0±0.9	28.4±1.0
halfcheetah-medium	43.2±0.6	55.2±27.8	-0.8±1.3	44.4	46.9±0.4	67.5±1.2	65.9±0.6
halfcheetah-expert	91.8±1.5	-0.8±1.8	4.1±5.7	104.8	97.3±1.1	105.2±2.6	106.8±3.4
halfcheetah-medium-expert	44.0±1.6	28.4±19.4	0.7±3.7	62.4	95.0±1.4	107.1±2.0	106.3±1.9
halfcheetah-medium-replay	37.6±2.1	0.8±1.0	6.6±11.0	46.2	45.3±0.3	63.9±0.8	61.3±1.9
halfcheetah-full-replay	62.9±0.8	86.8±1.0	27.8±35.4	-	76.9±0.9	84.5±1.2	84.6±0.9
hopper-random	3.7±0.6	9.9±1.5	3.4±2.2	10.8	5.3±0.6	31.3±0.0	25.3±10.4
hopper-medium	54.1±3.8	0.8±0.0	0.7±0.0	86.6	61.9±6.4	100.3±0.3	101.6±0.6
hopper-expert	107.7±9.7	0.7±0.0	0.8±0.0	109.9	106.5±9.1	110.3±0.3	110.1±0.1
hopper-medium-expert	53.9±4.7	0.7±0.0	0.8±0.0	111.0	96.9±15.1	110.1±0.3	110.7±0.1
hopper-medium-replay	16.6±4.8	7.4±0.5	27.5±15.2	48.6	86.3±7.3	101.8±0.5	101.0±0.5
hopper-full-replay	19.9±12.9	41.1±17.9	19.7±24.6	-	101.9±0.6	102.9±0.3	105.4±0.7
walker2d-random	1.3±0.1	0.9±0.8	6.9±8.3	7.0	5.4±1.7	21.7±0.0	16.6±7.0
walker2d-medium	70.9±11.0	-0.3±0.2	0.2±0.7	74.5	79.5±3.2	87.9±0.2	92.5±0.8
walker2d-expert	108.7±0.2	0.7±0.3	1.0±2.3	121.6	109.3±0.1	107.4±2.4	115.1±1.9
walker2d-medium-expert	90.1±13.2	1.9±3.9	-0.1±0.0	98.7	109.1±0.2	116.7±0.4	114.7±0.9
walker2d-medium-replay	20.3±9.8	-0.4±0.3	12.5±6.2	32.6	76.8±10.0	78.7±0.7	87.1±2.3
walker2d-full-replay	68.8±17.7	27.9±47.3	-0.2±0.3	-	94.2±1.9	94.6±0.5	99.8±0.7
Average	49.9	16.2	6.2	-	73.7	84.5	85.2

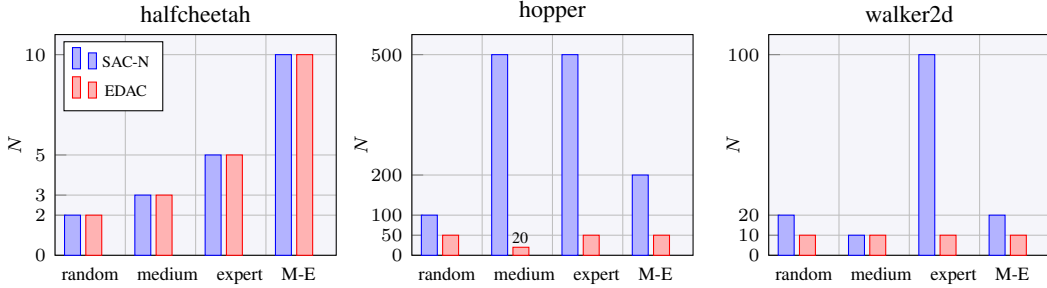


Figure 5: Minimum number of Q-ensembles (N) required to achieve the performance reported in Table 1. M-E denotes medium-expert. We omit the results of medium-replay and full-replay as SAC-N already works well with a small number of ensembles (less than or equal to 5). For more details of the experiment, please refer to Appendix C.

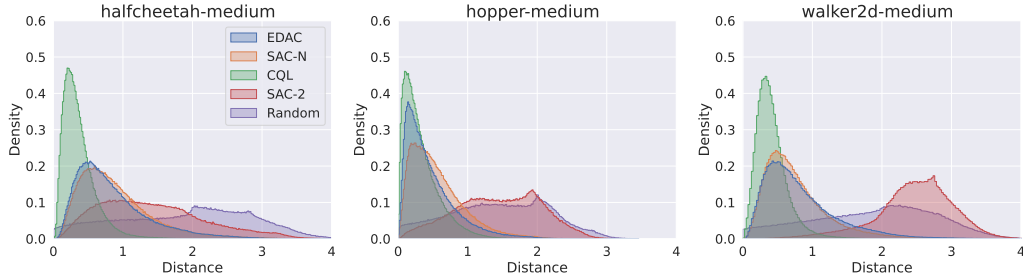


Figure 6: Histograms of the distances between the actions from each methods (EDAC, SAC-N, CQL, SAC-2, and a random policy) and the actions from the dataset. For more details of the experiment, please refer to Appendix C.

5.2 Evaluation on D4RL Adroit tasks

We also experiment on the more complex D4RL Adroit tasks that require controlling a 24-DoF robotic hand to perform tasks such as aligning a pen, hammering a nail, opening a door, or relocating a ball.

We use two types of datasets for each environment: *human*, containing 25 trajectories of human demonstrations, and *cloned*, a 50-50 mixture between the demonstration data and the behavioral cloned policy on the demonstrations. Note that for the Adroit tasks, we could not reproduce the CQL results from the paper completely. For the detailed procedure of reproducing the results of CQL, please refer to Appendix D.

Table 2: Normalized average returns on D4RL Adroit tasks, averaged over 4 random seeds.

Task Name	BC	SAC	REM	CQL (Paper)	CQL (Reproduced)	SAC- <i>N</i> (Ours)	EDAC (Ours)
pen-human	25.8±8.8	4.3±3.8	5.4±4.3	55.8	35.2±6.6	9.5±1.1	52.1±8.6
hammer-human	3.1±3.2	0.2±0.0	0.3±0.0	2.1	0.6±0.5	0.3±0.0	0.8±0.4
door-human	2.8±0.7	-0.3±0.0	-0.3±0.0	9.1	1.2±1.8	-0.3±0.0	10.7±6.8
relocate-human	0.0±0.0	-0.3±0.0	-0.3±0.0	0.35	0.0±0.0	-0.1±0.1	0.1±0.1
pen-cloned	38.3±11.9	-0.8±3.2	-1.0±0.1	40.3	27.2±11.3	64.1±8.7	68.2±7.3
hammer-cloned	0.7±0.3	0.1±0.1	-0.3±0.0	5.7	1.4±2.1	0.2±0.2	0.3±0.0
door-cloned	0.0±0.0	-0.3±0.1	-0.3±0.0	3.5	2.4±2.4	-0.3±0.0	9.6±8.3
relocate-cloned	0.1±0.0	-0.1±0.1	-0.2±0.2	-0.1	0.0±0.0	0.0±0.0	0.0±0.0

The evaluation results are summarized in Table 2. For pen-* tasks, where the considered algorithms achieve meaningful performance, EDAC outperforms or matches with the previous state-of-the-art. Especially, for pen-cloned, both EDAC and SAC-*N* achieve 75% higher score compared to CQL. Unlike the results from the Gym tasks, we find that SAC-*N* falls behind in some datasets, for example, pen-human, which could in part due to the size of the dataset being exceptionally small (5000 transitions). However, our method with ensemble diversification successfully overcomes this difficulty.

5.3 Computational cost comparison

We compared the computational cost of our methods with vanilla SAC and CQL on hopper-medium-v2, where our methods require the largest number of Q-networks. For each method, we measure the runtime per training epoch (1000 gradient steps) along with GPU memory consumption. We run our experiments on a single machine with one RTX 3090 GPU and provide the results in Table 3.

As the result shows, our method EDAC runs faster than CQL with comparable memory consumption. Note that CQL is about twice as slower than vanilla SAC due to the additional computations for Q-value regularization (e.g., dual update and approximate logsumexp via sampling). Meanwhile, the inference to the Q-network ensemble in SAC-*N* and EDAC is embarrassingly parallelizable, minimizing the runtime increase with the number of Q-networks. Also, we emphasize that our gradient diversification term in Equation (4) has linear computational complexity, as we can reformulate the term using the sum of the gradients.

Table 3: Computational costs of each method.

	Runtime (s/epoch)	GPU Mem. (GB)
SAC	21.4	1.3
CQL	38.2	1.4
SAC-500	44.1	5.1
EDAC	30.8	1.8

6 Related Works

Model-free offline RL A popular approach for offline RL is to regularize the learned policy to be close to the behavior policy where the offline dataset was collected. BCQ [11] uses a generative model to produce actions with high similarity to the dataset and trains a restricted policy to choose the best action from the neighborhood of the generated actions. Another line of work, such as BEAR [15] or BRAC [28], stabilizes policy learning by penalizing the divergence from the dataset measured by KL divergence or MMD. While these policy-constraint methods demonstrate high performance on datasets from expert behavior policies, they fail to find optimal policies from datasets with suboptimal policies due to the strict policy constraints [9]. Also, these methods require an accurate estimation of the behavior policy, which might be difficult in complex settings with multiple behavior sources or high-dimensional environments. To address these issues, CQL [16] directly regularizes Q-functions

by introducing a term that minimizes the Q-values for out-of-distribution actions and maximizes the Q-values for in-distribution actions. Without such explicit regularizations, REM [2] proposes to use a random convex combination of Q-network ensembles on environments with discrete action spaces [4].

Estimation bias in Q-learning While Q-learning is one of the most popular algorithms in reinforcement learning, it suffers from overestimation bias due to the maximum operation $\max_{\mathbf{a}' \in \mathcal{A}} Q(\mathbf{s}', \mathbf{a}')$ used during Q-function updates [10, 25]. This overestimation bias, together with the bootstrapping, can lead to a catastrophic build-up of errors during the Q-learning process. To resolve this issue, TD3 [10] introduces a clipped version of Double Q-learning [25] that takes the minimum value of two critics. Subsequently, Maxmin Q-learning [17] theoretically shows that the overestimation bias can be controlled by the number of ensembles in the clipped Q-learning. The overestimation problem in Q-learning can be exacerbated in the offline setting since the extrapolation error cannot be corrected with further interactions with the environment, and existing offline RL algorithms handle the bias by introducing constrained policy optimization [11, 15] or conservative Q-learning frameworks [16].

Uncertainty measures in RL Uncertainty estimates have been widely used in RL for various purposes including exploration, Q-learning, and planning. Bootstrapped DQN [21] leverages an ensemble of Q-functions to quantify the uncertainty of the Q-value, and utilizes it for efficient exploration. Following this work, the UCB exploration algorithm [5] constructs an upper confidence bound [3] of the Q-values using the empirical mean and standard deviation of Q-ensembles, which is used to promote efficient exploration by applying the principle of optimism in the face of uncertainty [7]. Osband et al. [22] proposes a randomly initialized Q-ensemble that reflects the concept of prior functions in Bayesian inference and Abbas et al. [1] introduces an uncertainty incorporated planning with imperfect models. The notion of uncertainty has also been considered in offline RL, mostly in the framework of model-based offline RL. Especially, MOPO [30] and MOREL [13] measure the uncertainty of the model’s prediction to formulate an uncertainty-penalized policy optimization problem in the offline RL setting. These methods introduce an ensemble of dynamics models for the quantification of the uncertainty, whereas our work adopts an ensemble of Q-functions for uncertainty-aware Q-learning.

7 Conclusion

We have shown that clipped Q-learning can be efficiently leveraged to construct an uncertainty-based offline RL method that outperforms previous methods on various datasets. Based on this observation, we proposed Ensemble-Diversifying Actor-Critic (EDAC) that effectively reduces the required number of ensemble networks for quantifying and penalizing the epistemic uncertainty. Our method does not require any explicit estimation of the data collecting policy or sampling from the out-of-distribution data and respects the epistemic uncertainty of each data point during penalization. EDAC, while requiring up to 90% less number of ensemble networks compared to the vanilla Q-ensemble, exhibits state-of-the-art performance on various datasets.

Acknowledgements

This work was supported in part by Samsung Advanced Institute of Technology, Samsung Electronics Co., Ltd., 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 No. 2019-0-01371, Development of brain-inspired AI with human-like intelligence), and Research Resettlement Fund for the new faculty of Seoul National University. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA2386-20-1-4043.

References

- [1] Zaheer Abbas, Samuel Sokota, Erin Talvitie, and Martha White. Selective dyna-style planning under limited model capacity. In *ICML*, 2020.

- [2] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020.
- [3] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902, 2009.
- [4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.
- [5] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.
- [6] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *ICML*, 2019.
- [7] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor-critic. In *NeurIPS*, 2019.
- [8] William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.
- [9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.
- [10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.
- [11] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.
- [12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.
- [13] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. In *NeurIPS*, 2020.
- [14] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *NeurIPS*, 2000.
- [15] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, 2019.
- [16] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, 2020.
- [17] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *ICLR*, 2020.
- [18] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *ICML*, 2021.
- [19] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- [20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [21] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *NeurIPS*, 2016.
- [22] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, 2018.

- [23] JP Royston. Expected normal order statistics(exact and approximate). *Applied Statistics*, 31(2): 161–5, 1982.
- [24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.
- [25] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.
- [26] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojciech M Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, et al. Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind blog*, 2, 2019.
- [27] David S Watkins. Understanding the qr algorithm. *SIAM review*, 24(4):427–440, 1982.
- [28] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.
- [29] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua M. Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning, 2021.
- [30] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. In *NeurIPS*, 2020.

A Ensemble gradient diversification

A.1 Proofs

Lemma 1. *The total variance of the matrix $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ is equal to $1 - \|\bar{q}\|_2^2$, where $\bar{q} = \frac{1}{N} \sum_{j=1}^N \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$.*

Proof. For simplicity, we denote $\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$ by q_j and their average by $\bar{q} = \frac{1}{N} \sum_j q_j$. Then, the total variance of the matrix, which is equivalent to the trace of the matrix by definition, formulates as below:

$$\begin{aligned}
\text{tr}(\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))) &= \text{tr}\left(\frac{1}{N} \sum_{j=1}^N (q_j - \bar{q})(q_j - \bar{q})^\top\right) \\
&= \frac{1}{N} \sum_{j=1}^N \text{tr}((q_j - \bar{q})(q_j - \bar{q})^\top) \\
&= \frac{1}{N} \sum_{j=1}^N \text{tr}((q_j - \bar{q})^\top (q_j - \bar{q})) && (\text{tr}(AB) = \text{tr}(BA)) \\
&= \frac{1}{N} \sum_{j=1}^N (q_j - \bar{q})^\top (q_j - \bar{q}) \\
&= \frac{1}{N} \sum_{j=1}^N (q_j^\top q_j - 2q_j^\top \bar{q} + \bar{q}^\top \bar{q}) \\
&= 1 - 2 \left(\frac{1}{N} \sum_{j=1}^N q_j \right)^\top \bar{q} + \bar{q}^\top \bar{q} && (\|q_j\|_2 = 1) \\
&= 1 - \bar{q}^\top \bar{q} \\
&= 1 - \|\bar{q}\|_2^2.
\end{aligned}$$

□

Proposition 1. *Suppose $Q_{\phi_j}(\mathbf{s}, \mathbf{a}) = Q(\mathbf{s}, \mathbf{a})$ and $Q_{\phi_j}(\mathbf{s}, \cdot)$ is locally linear in the neighborhood of $\mathbf{a}$ for all $j \in [N]$. Let $\lambda_{\min}$ and $\mathbf{w}_{\min}$ be the smallest eigenvalue and the corresponding normalized eigenvector of the matrix $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ and $\epsilon > 0$ be the value such that $\min_{i \neq j} \langle \nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}), \nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}) \rangle = 1 - \epsilon$. Then, the variance of the Q -values for an OOD action in the neighborhood along the direction of $\mathbf{w}_{\min}$ is upper-bounded as follows:*

$$\text{Var}(Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w}_{\min})) \leq \frac{1}{|\mathcal{A}|} \frac{N-1}{N} k^2 \epsilon,$$

where $|\mathcal{A}|$ is the action space dimension.

Proof. We first prove that the smallest eigenvalue $\lambda_{\min}$ of $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$ is upper-bounded by some constant multiple of ϵ . For simplicity, we denote $\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})$ by q_j and their average by

$\bar{q} = \frac{1}{N} \sum_j q_j$. We first compute the norm of the average of the gradients, which can be expressed by

$$\begin{aligned}
\|\bar{q}\|_2^2 &= \langle \bar{q}, \bar{q} \rangle \\
&= \left\langle \frac{1}{N} \sum_{i=1}^N q_i, \frac{1}{N} \sum_{j=1}^N q_j \right\rangle \\
&= \frac{1}{N^2} \sum_{1 \leq i, j \leq N} \langle q_i, q_j \rangle \\
&= \frac{1}{N^2} \left(\sum_{j=1}^N \langle q_j, q_j \rangle + \sum_{1 \leq i \neq j \leq N} \langle q_i, q_j \rangle \right) \\
&\geq \frac{1}{N^2} (N + N(N-1)(1-\epsilon)) \\
&= 1 - \frac{(N-1)}{N} \epsilon.
\end{aligned}$$

By Lemma 1, the total variance of the matrix is less or equal to $\frac{N-1}{N} \epsilon$. Using the fact that the total variance is equivalent to the sum of the eigenvalues and the eigenvalues of a variance matrix is non-negative, we have

$$\begin{aligned}
\lambda_{\min} &\leq \frac{1}{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} \lambda_j \\
&= \frac{1}{|\mathcal{A}|} \text{tr}(\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))) \\
&\leq \frac{1}{|\mathcal{A}|} \frac{N-1}{N} \epsilon,
\end{aligned} \tag{5}$$

where $\lambda_1, \dots, \lambda_{|\mathcal{A}|}$ are the eigenvalues of $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a}))$.

Note that, using the fact that the Q-values coincide at the action $\mathbf{a}$ and the local linearity of the Q-functions, we have derived

$$\text{Var}(Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w})) = k^2 \mathbf{w}^\top \text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})) \mathbf{w}. \tag{6}$$

Plugging $\mathbf{w} = \mathbf{w}_{\min}$ in Equation (6) and using Equation (5), we have

$$\begin{aligned}
\text{Var}(Q_{\phi_j}(\mathbf{s}, \mathbf{a} + k\mathbf{w}_{\min})) &= k^2 \mathbf{w}_{\min}^\top \text{Var}(\nabla_{\mathbf{a}} Q_{\phi_j}(\mathbf{s}, \mathbf{a})) \mathbf{w}_{\min} \\
&= k^2 \lambda_{\min} \\
&\leq \frac{1}{|\mathcal{A}|} \frac{N-1}{N} k^2 \epsilon.
\end{aligned}$$

□

A.2 Relationship between maximizing the total variance and maximizing the smallest eigenvalue

As we have shown in Section 4, maximizing the total variance of the matrix $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}))$ is equivalent to minimizing the cosine similarity of all distinct pairs of the gradients $\nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a})$, which makes the gradients uniformly distributed on the unit sphere $S^{|\mathcal{A}|-1}$. Therefore, if the trace is sufficiently maximized, then we can see $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}))$ as a sample variance matrix of a uniform spherical distribution. It can be easily proved that the variance matrix of a uniform distribution on is $\frac{1}{|\mathcal{A}|} I$, whose all eigenvalues are equal to $\frac{1}{|\mathcal{A}|}$, by Proposition 2.

Proposition 2. *The variance matrix of the uniform spherical distribution $X \sim \mathcal{U}(S^{n-1})$ is $\frac{1}{n} I$.*

Proof. Let $X = (X_1, \dots, X_n)$. Then $X_{-i} = (X_1, \dots, -X_i, \dots, X_n)$ is also from the uniform spherical distribution. Therefore, we have $\mathbb{E}[X_i] = \mathbb{E}[-X_i] = 0$ and $\mathbb{E}[X_i X_j] = \mathbb{E}[-X_i X_j] =$

0, $\forall i \neq j$. For the diagonal entries of the variance matrix, we have $\mathbb{E}[\sum_{i=1}^n X_i^2] = \sum_{i=1}^n \mathbb{E}[X_i^2] = 1$ by the definition of the spherical distribution and $\mathbb{E}[X_i^2] = \mathbb{E}[X_j^2]$ by the symmetry of the distribution. Therefore, we have $\mathbb{E}[X_i^2] = \frac{1}{n}$ and $\text{Var}(X) = \frac{1}{n}I$. $\square$

Note that the smallest eigenvalue of $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}))$ is less or equal to $\frac{1}{|A|}$, since the total variance is upper-bounded by 1 due to Lemma 1. Therefore, as the number of Q-ensembles goes to infinity, $\text{Var}(\nabla_{\mathbf{a}} Q_{\phi_i}(\mathbf{s}, \mathbf{a}))$ converges to $\frac{1}{|A|}I$, attaining the maximum value for the smallest eigenvalue.

B Implementation details

SAC We use the SAC implementation from rlkit³. We use its default parameters except for increasing the number of layers for both the policy network and the Q-function networks from 2 to 3, following the protocol of CQL.

REM We implement a continuous control version of REM on top of SAC by modifying the Bellman residual term to

$$\min_{\phi} \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}' \sim \mathcal{D}, \xi \sim P_{\Delta}} \left[\left(\sum_{j=1}^N \xi_j Q_{\phi_j}(\mathbf{s}, \mathbf{a}) - \left(r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_{\theta}(\cdot | \mathbf{s}')} \left[\sum_{j=1}^N \xi_j Q_{\phi_j}(\mathbf{s}', \mathbf{a}') \right] \right) \right)^2 \right],$$

where P_{Δ} represents a probability distribution over the standard $(N - 1)$ -simplex $\Delta^{N-1} = \{\xi \in \mathbb{R}^N : \xi_1 + \xi_2 + \dots + \xi_N = 1, \xi_n \geq 0, n = 1, \dots, N\}$. Following the original REM paper, we use a simple probability distribution: $\xi_n = \xi'_n / \sum_k \xi'_k$, where $\xi'_k \sim U(0, 1)$ for $k = 1, \dots, N$. For a fair comparison with our ensemble algorithms, we sweep the ensemble size N within $\{2, 5, 10, 20, 50, 100, 200, 500, 1000\}$ and report the best number.

CQL We use the official implementation by the authors⁴. For MuJoCo Gym tasks, the recommended hyperparameters in the codebase differ from the original paper due to the updates in the D4RL datasets. We tried both versions of hyperparameter settings and found the codebase version outperforms the paper version while matching the numbers in the paper reasonably well. Therefore, we follow the guidelines from the official code and use the fixed α version, searching for the parameters within $\alpha \in \{5, 10\}$ and policy learning rate $\in \{1e - 4, 3e - 4\}$. We chose $\alpha = 10.0$ with policy learning rate $= 1e - 4$ as the default as it gives the best results in most of the datasets. However, we use the dual gradient descent version with $\tau = 10.0$ and policy learning rate $= 1e - 4$ on some datasets, such as halfcheetah-random, since the fixed α version could not reproduce the results from the paper on those datasets. For the Adroit tasks, the codebase does not provide separate guidelines, and we use the hyperparameters listed in the paper.

SAC- N (Ours) We keep the default setting from the SAC experiments other than the ensemble size N . On halfcheetah and walker2d environments, we tune N in the range of $\{5, 10, 20\}$ except for walker2d-expert, which requires up to $N = 100$. For hopper, we tune within $N \in \{100, 200, 500, 1000\}$. The hyperparameters selected are listed in Table 4. As we noted in Figure 5, some datasets can be dealt with less N (e.g., *-replay). However, we tried to keep the hyperparameters within an environment consistent in order to reduce hyperparameter sensitivity. For Adroit tasks, we sweep N in the range of $\{20, 50, 100, 200\}$ and report the selected N in Table 5.

EDAC (Ours) For Mujoco Gym tasks, we tune the ensemble size N within the range of $\{10, 20, 50\}$ and the weight of the ensemble gradient diversity term η within $\{0.0, 1.0, 5.0\}$. Note that we use the same N on each environment. For Adroit tasks, we sweep the parameters on $N \in \{20, 50, 100\}$ and $\eta \in \{100, 200, 500, 1000\}$ except for pen-cloned, which uses $\eta = 10.0$. While we can also achieve competitive performance on pen-cloned with larger η , we found lower η helps to mitigate the performance degradation on further training steps. The selected hyperparameters on each environment are listed in Table 4 and Table 5, respectively.

³<https://github.com/vitchyr/rlkit>

⁴<https://github.com/aviralkumar2907/CQL>

Table 4: Hyperparameters used in the D4RL MuJoCo Gym experiments.

Task Name	SAC- N (N)	EDAC (N, η)
halfcheetah-random	10	10, 0.0
halfcheetah-medium	10	10, 1.0
halfcheetah-expert	10	10, 1.0
halfcheetah-medium-expert	10	10, 5.0
halfcheetah-medium-replay	10	10, 1.0
halfcheetah-full-replay	10	10, 1.0
hopper-random	500	50, 0.0
hopper-medium	500	50, 1.0
hopper-expert	500	50, 1.0
hopper-medium-expert	200	50, 1.0
hopper-medium-replay	200	50, 1.0
hopper-full-replay	200	50, 1.0
walker2d-random	20	10, 1.0
walker2d-medium	20	10, 1.0
walker2d-expert	100	10, 5.0
walker2d-medium-expert	20	10, 5.0
walker2d-medium-replay	20	10, 1.0
walker2d-full-replay	20	10, 1.0

Table 5: Hyperparameters used in the D4RL Adroit experiments.

Task Name	SAC- N (N)	EDAC (N, η)
pen-human	100	20, 1000.0
pen-cloned	100	20, 10.0
hammer-human	100	50, 200.0
hammer-cloned	100	50, 200.0
door-human	100	50, 200.0
door-cloned	100	50, 200.0
relocate-human	100	50, 200.0
relocate-cloned	100	50, 200.0

C Experimental settings

MuJoCo Gym We use the v2 version of each dataset (*e.g.*, halfcheetah-random-v2) which fixes some of the bugs from the previous versions. We run each algorithm for 3 million training steps and report the normalized average return of each policy. While the CQL paper originally used 1 million steps, we found increasing this to 3 million helps the algorithms to converge on more complex datasets such as *-medium-expert.

Adroit We use the v1 version of each dataset. On these datasets, we adopt max Q backup from CQL and normalize the rewards for training stability. As we will discuss in Appendix D, the performance of the baseline algorithm CQL degrades after some steps of training. Therefore, for a fair comparison, we run each algorithm for 200,000 steps and report the normalized average return.

Minimum required Q-ensembles (Figure 5) To check the minimum required number of Q-ensembles for each dataset, we sweep N within the range of $\{2, 3, 5, 10, 20, 50, 100, 200, 500, 1000\}$ and report the minimum N that achieves the performance similar to Table 1. We find EDAC successes to reduce the required N significantly when the original requirement is high (*e.g.*, hopper, walker2d-expert).

Action distance histograms (Figure 6) To draw the histogram, we sample 500,000 random (s, a) pairs from each dataset and measure the ℓ_2 distance between the action sampled from each policy after full training and the dataset action.

D Reproducing CQL in Adroit

Since the pen-* tasks are where the considered algorithms show meaningful performance, we focused on reproducing the reported results for those tasks. After running CQL with the parameters given in the original paper, we found that the performance of CQL degrades after about 200,000 steps, as shown in Figure 7. While we are not sure of the cause of this performance gap, it could be due to the difference in the `min_q_weight` parameter setting, which was not specified in the original paper, or a minor modification we applied to the code to fix the backpropagation issue⁵. Meanwhile, for a fair comparison, on Adroit we chose to use early-stopping and train each algorithm for 200,000 steps. Also, we include the reported CQL numbers for all experiments.

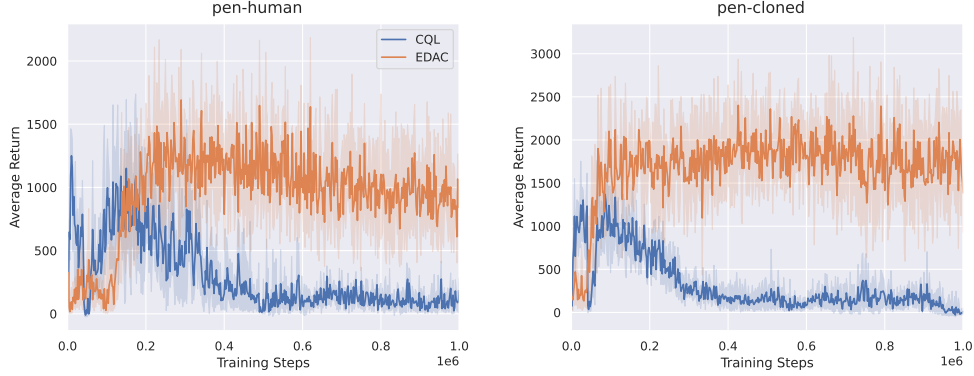


Figure 7: Performance of EDAC and CQL on pen-* datasets. ‘Average Return’ denotes the undiscounted return of each policy on evaluation. Results averaged over 4 seeds.

E Comparison with more baselines

We additionally compared our methods with more baselines on D4RL Gym datasets. First, we add comparisons with some of the well-known offline RL methods, BCQ [11], BEAR [15], BRAC [28], and MOREL [13]. Also, we include the results of UWAC [29], a concurrent work that also utilizes Q-value uncertainty. We reproduced all the methods by following the hyperparameter search procedure listed in each paper and selected the best results. We report the normalized average return results in Table 6.

Table 6: Extended results of normalized average returns on D4RL Gym tasks, averaged over 4 random seeds. CQL (Paper) denotes the results reported in the original paper.

Task Name	BC	SAC	REM	BCQ	BEAR	BRAC	MOREL	UWAC	CQL (Paper)	CQL (Reproduced)	SAC-N (Ours)	EDAC (Ours)
halfcheetah-random	2.2±0.0	29.7±1.4	-0.8±1.1	2.2±0.0	12.6±1.0	24.3±0.7	38.9±1.8	2.3±0.0	35.4	31.3±3.5	28.0±0.9	28.4±1.0
halfcheetah-medium	43.2±0.6	55.2±27.8	-0.8±1.3	46.6±0.4	42.8±0.1	51.9±0.3	60.7±4.4	43.7±0.4	44.4	46.9±0.4	67.5±1.2	65.9±0.6
halfcheetah-expert	91.8±1.5	-0.8±1.8	4.1±5.7	89.9±9.6	92.6±0.6	39.0±13.8	8.4±11.8	94.7±1.1	104.8	97.3±1.1	105.2±2.6	106.8±3.4
halfcheetah-medium-expert	44.0±1.6	28.4±19.4	0.7±3.7	95.4±2.0	45.7±4.2	52.3±0.1	80.4±11.7	47.0±6.0	62.4	95.0±1.4	107.1±2.0	106.3±1.9
halfcheetah-medium-replay	37.6±2.1	0.8±1.0	6.6±11.0	42.2±0.9	39.4±0.8	48.6±0.4	44.5±5.6	38.9±1.1	46.2	45.3±0.3	63.9±0.8	61.3±1.9
halfcheetah-full-replay	62.9±0.8	86.8±1.0	27.8±35.4	69.5±4.0	60.1±3.2	78.0±0.7	70.1±5.1	65.1±0.5	-	76.9±0.9	84.5±1.2	84.6±0.9
hopper-random	3.7±0.6	9.9±1.5	3.4±2.2	7.8±0.6	3.6±3.6	8.1±0.6	38.1±10.1	2.6±0.3	10.8	5.3±0.6	31.3±0.0	25.3±10.4
hopper-medium	54.1±3.8	0.8±0.0	0.7±0.0	59.4±8.3	55.3±3.2	77.8±6.1	84.0±17.0	52.6±4.0	86.6	61.9±6.4	100.3±0.3	101.6±0.6
hopper-expert	107.7±9.7	0.7±0.0	0.8±0.0	109±4.0	39.4±20.5	78.1±52.3	80.4±34.9	111.0±0.8	109.9	106.5±9.1	110.3±0.3	110.1±0.1
hopper-medium-expert	53.9±4.7	0.7±0.0	0.8±0.0	106.9±5.0	66.2±8.5	81.3±8.0	105.6±8.2	54.8±3.2	111.0	96.9±15.1	110.1±0.3	110.7±0.1
hopper-medium-replay	16.6±4.8	7.4±0.5	27.5±15.2	60.9±14.7	57.7±16.5	62.7±30.4	81.8±17.0	31.1±14.8	48.6	86.3±7.3	101.8±0.5	101.0±0.5
hopper-full-replay	19.9±12.9	41.1±17.9	19.7±24.6	46.6±13.0	54.0±24.0	107.4±0.5	94.4±20.5	21.9±8.4	-	101.9±0.6	102.9±0.3	105.4±0.7
walker2d-random	1.3±0.1	0.9±0.8	6.9±8.3	4.9±0.1	4.3±1.2	1.3±1.4	16.0±7.7	1.5±0.3	7.0	5.4±1.7	21.7±0.0	16.6±7.0
walker2d-medium	70.9±11.0	-0.3±0.2	0.2±0.7	71.8±7.2	59.8±40.0	59.7±39.9	72.8±11.9	66.0±9.0	74.5	79.5±3.2	87.9±0.2	92.5±0.8
walker2d-expert	108.7±0.2	0.7±0.3	1.0±2.3	106.3±5.0	110.1±0.6	55.2±62.2	62.6±29.9	108.4±0.5	121.6	109.3±0.1	107.4±2.4	115.1±1.9
walker2d-medium-expert	90.1±13.2	1.9±3.9	-0.1±0.0	107.7±3.8	107.0±2.9	9.3±18.9	107.5±5.6	85.7±14.0	98.7	109.1±0.2	116.7±0.4	114.7±0.9
walker2d-medium-replay	20.3±9.8	-0.4±0.3	12.5±6.2	57.0±9.6	12.2±4.7	40.1±47.9	40.8±20.4	27.1±9.6	32.6	76.8±10.0	78.7±0.7	87.1±2.3
walker2d-full-replay	68.8±17.7	27.9±47.3	-0.2±0.3	71.0±21.8	79.6±15.6	96.9±2.2	84.8±13.1	60.7±15.6	-	94.2±1.9	94.6±0.5	99.8±0.7
Average	49.9	16.2	6.2	64.2	52.4	54.0	65.1	50.8	-	73.7	84.5	85.2

The results show our methods outperform all the baseline methods on most of the datasets considered. Also, we reiterate that while the performance of EDAC is marginally better than SAC-N, EDAC achieves this result with a much smaller Q-ensemble size.

⁵<https://github.com/aviralakumar2907/CQL/issues/5>

F CQL with N Q-networks

Since other offline RL methods may also benefit from larger N or ensemble diversification, here we evaluate CQL- N and CQL with ensemble diversification for ablation. For CQL- N , we tried $N \in \{2, 5, 10, 50, 100\}$, where $N = 2$ denotes the original version of CQL. For CQL with ensemble diversification, we added our diversification term to the CQL loss function and swept the coefficient η in the range of $\{0.5, 1.0, 5.0\}$, which is the same range used in EDAC. The normalized return evaluation results on D4RL Gym *-medium datasets are shown in Table 7.

Table 7: Performance of CQL with N Q-networks on D4RL Gym *-medium datasets.

		halfcheetah-medium	hopper-medium	walker2-medium
CQL-N	$N = 2$	46.9 $\pm$ 0.4	61.9 $\pm$ 6.4	79.5 $\pm$ 3.2
	$N = 5$	47.1 $\pm$ 0.3	61.6 $\pm$ 6.0	80.8 $\pm$ 4.9
	$N = 10$	45.9 $\pm$ 0.3	60.1 $\pm$ 4.8	70.9 $\pm$ 0.9
	$N = 50$	44.2 $\pm$ 0.4	54.3 $\pm$ 2.0	69.4 $\pm$ 0.0
	$N = 100$	43.7 $\pm$ 0.2	43.7 $\pm$ 0.8	71.3 $\pm$ 3.8
CQL w/ diversification	$\eta = 0.5$	46.5 $\pm$ 0.4	65.8 $\pm$ 11.2	82.2 $\pm$ 0.6
	$\eta = 1.0$	47.2 $\pm$ 0.1	69.2 $\pm$ 8.8	80.5 $\pm$ 3.1
	$\eta = 5.0$	47.4 $\pm$ 0.5	60.9 $\pm$ 3.2	82.1 $\pm$ 1.2
SAC-N		67.5$\pm$1.2	100.3$\pm$0.3	87.9$\pm$0.2
EDAC		65.9$\pm$1.6	101.6$\pm$0.6	92.5$\pm$0.8

We observe that even though increasing the number of Q-networks or applying gradient diversification do help CQL on some of the datasets, the improved performance still falls far behind our methods (SAC- N , EDAC).

G Comparison to variance regularization

In this section, we compare EDAC with increasing the variance of the Q-estimates for in-distribution actions, which is another possible option for ensemble diversification. Table 8 shows the average return and the Q-value estimation statistics on the walker2d-expert dataset when using the Q-estimate variance regularizer, compared to EDAC. *Var reg* adds to SAC- N a regularizing term that explicitly increases the variance of the Q-estimates, weighted by a coefficient c . *Q Avg* denotes the estimated Q-values of each model in evaluation. *Q Std* means the standard deviation of Q-estimates from an ensemble on the given actions. *Q Std gap* means the gap of standard deviations from behavior and random actions.

Table 8: Comparison of EDAC with Q-estimate variance regularization on walker2d-expert dataset. Same number of Q-networks is used for all methods.

		Return	Q Avg	Q Std (behavior action)	Q Std (random action)	Q Std gap
Var reg	$c = 50$	511	overflow	N/A	N/A	N/A
	$c = 100$	20	-95	5.3	7	1.7
	$c = 200$	368	-929	10.6	15.1	4.5
EDAC		5236	392	1.2	10.5	9.3

On the walker2d-expert dataset, adding the variance-enhancing regularizer either leads to two results: (1) Exploding Q-values when the regularization is not strong ($c = 50$) or (2) severe Q-value underestimation when the regularization is stronger ($c = 100, 200$). The reason behind these two extreme modes is that the gap of the Q-estimate variance between behavior actions and OOD actions, which is crucial for conservative learning, increases much slower than the absolute increase of the Q-estimate variances. For example, on $c = 200$, the Q-estimate Std gap is 4.5. This gap is about half of EDAC, whereas the absolute Q-estimate Stds on both actions are much higher. In

EDAC, the variance of Q-estimates on behavior actions remains small even though the OOD actions are sufficiently penalized, as we only diversify the Q-networks’ gradients instead of the Q-values themselves.

H Hyperparameter sensitivity

To measure the hyperparameter sensitivity of EDAC, we sweep the weight of the gradient diversification term η in the range of $\{0.0, 0.5, 1.0, 2.0, 5.0\}$ on the hopper datasets, fixing the number of Q-networks to $N = 50$, and present the results in Table 9.

Table 9: Performance of EDAC over various η on the D4RL Gym hopper datasets.

Dataset type	$\eta = 0.0$	$\eta = 0.5$	$\eta = 1.0$	$\eta = 2.0$	$\eta = 5.0$
random	25.3±10.4	9.3±6.2	6.7±0.8	3.8±1.5	1.9±0.8
medium	7.3±0.1	102.2±0.4	101.6±0.5	94.5±12.4	75.5±24.1
expert	2.3±0.1	110.3±0.2	110.1±0.1	109.8±0.2	109.9±0.2
medium-expert	46.9±33.0	103.8±12.4	110.7±0.1	109.8±0.2	109.8±0.2
medium-replay	100.9±0.4	100.3±0.8	101.0±0.4	100.2±0.5	20.6±0.7
full-replay	105.6±0.4	104.9±0.5	4105.4±0.6	104.0±0.2	106.3±0.9

The results show that except for the random dataset, there exists a large well of hyperparameters where EDAC achieves expert-level performance. We also observe that increasing η sometimes degrades the performance on random, medium, and medium-replay datasets which contain trajectories drawn from suboptimal policies. Intuitively, the gradient diversification term induces the learned policy to favor in-distribution actions over OOD actions. Therefore, increasing η can lead to a more conservative policy, which is undesirable if the behavior policy is suboptimal.

Preemptive Image Robustification for Protecting Users against Man-in-the-Middle Adversarial Attacks

Seungyong Moon^{*1}, Gaon An^{*1}, Hyun Oh Song^{†12}

¹ Department of Computer Science and Engineering, Seoul National University, Seoul, Korea

² DeepMetrics, Seoul, Korea

{symoon11, white0234, hyunoh}@mllab.snu.ac.kr

Abstract

Deep neural networks have become the driving force of modern image recognition systems. However, the vulnerability of neural networks against adversarial attacks poses a serious threat to the people affected by these systems. In this paper, we focus on a real-world threat model where a Man-in-the-Middle adversary maliciously intercepts and perturbs images web users upload online. This type of attack can raise severe ethical concerns on top of simple performance degradation. To prevent this attack, we devise a novel bi-level optimization algorithm that finds points in the vicinity of natural images that are robust to adversarial perturbations. Experiments on CIFAR-10 and ImageNet show our method can effectively robustify natural images within the given modification budget. We also show the proposed method can improve robustness when jointly used with randomized smoothing.

1 Introduction

Recent progress in deep neural networks has enabled substantial performance gains in various computer vision tasks, including image classification, object detection, and semantic segmentation. Leveraging this advance, more practitioners are deploying neural network-based image recognition systems in real-world applications, such as image tagging or face recognition. However, neural networks are vulnerable to *adversarial examples* (Szegedy et al. 2013), minute input perturbations intentionally designed to mislead networks to yield incorrect predictions. These adversarial examples can significantly degrade the performance of the network models, raising security concerns about their deployment.

When these image recognition systems are deployed to applications where users freely upload images from local machines to remote storage, such as social media, this vulnerability can pose another serious threat, especially to the individual application users. Consider there exists a man-in-the-middle (MitM) adversary that can intercept and add perturbations to the images web users upload during transmission (Figure 1). Then, this adversary can easily vandalize neural network-based web services such as image auto-tagging

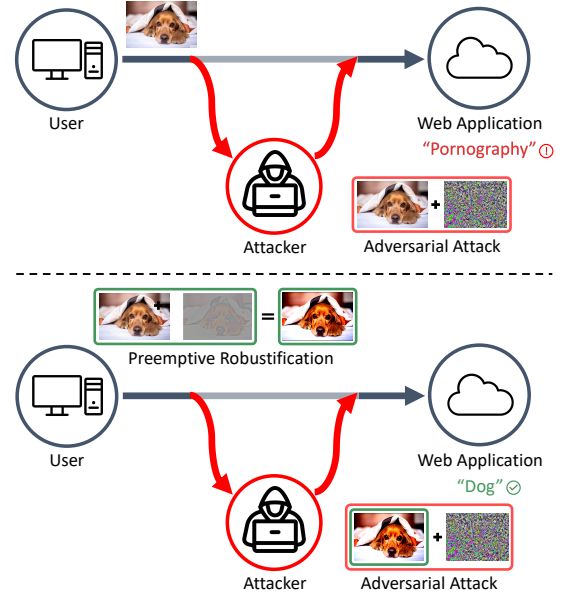


Figure 1: Illustration of our proposed method. Without protection, a MitM adversary can easily perturb the user’s image to be misclassified by the web application (top). Our proposed method preemptively robustifies the user’s image, securing the image from adversarial attack (bottom).

in social media apps by perturbing the images to be misclassified. This type of attack can severely deteriorate user experience, especially as the adversary can further use this attack to insult the uploaders beyond simple misclassification. Even though the MitM attack is one of the most lethal cyber threats (Desmedt 2011; Li et al. 2019; Wang et al. 2019), protecting neural networks from this type of attack has been much less studied in adversarial machine learning literature.

In this work, we develop a new defense framework to protect web users from MitM attacks. To provide more effective protection measures for the users, we focus on the fact that users hold control of their images before the adversary, unlike in conventional adversarial attack scenarios. Based on this observation, we ask the following question:

- *Can we preemptively manipulate images slightly to be robust against MitM adversarial attacks?*

^{*}These authors contributed equally.

[†]Corresponding author

To answer this, we explore the existence of points in image space that are resistant to adversarial perturbations, given a trained classifier. We propose a novel bi-level optimization algorithm for finding those robust points under a given modification budget starting from natural images and measure the degree of robustness achievable by utilizing these points, which we denote as *preemptive robustness*. Moreover, we propose a new network training scheme that further improves preemptive robustness. We validate the effectiveness of our proposed framework in the image classification task on CIFAR-10 (Krizhevsky and Hinton 2009) and ImageNet (Russakovsky et al. 2015) datasets. Our extensive experiments demonstrate that our framework can successfully robustify images against a wide range of adversarial attacks in both black-box and white-box settings. We also observe that our method can enhance the preemptive robustness on smoothed classifiers.

In summary, our main contributions are as follows:

- We introduce a novel real-world adversarial attack scenario targeting users in image recognition systems.
- We propose a new defense framework to improve preemptive robustness and formulate the preemptive robustification process as a bi-level optimization problem.
- We demonstrate our proposed framework can significantly improve preemptive robustness against a wide range of adversarial attacks on standard benchmarks.

2 Related Works

Adversarial robustness of neural networks Most prior work on adversarial robustness aims to train neural networks that achieve high accuracies on adversarially perturbed inputs. PGD adversarial training improves the empirical robustness of neural networks by augmenting training data with multi-step PGD adversarial examples (Madry et al. 2017). Some recent works report performance gains over PGD adversarial training by modifying the adversarial example generation procedure (Zhang and Wang 2019; Zhang et al. 2020). However, most of the recent algorithmic improvements can be matched by simply using early stopping with PGD adversarial training (Rice, Wong, and Kolter 2020; Croce and Hein 2020). Despite its effectiveness, a major drawback of adversarial training is that it takes a huge computational cost to generate adversarial examples during training. To address this, several works develop fast adversarial training methods by reusing the gradient computation or reducing the number of attack iterations (Shafahi et al. 2019; Zhang et al. 2019a; Wong, Rice, and Kolter 2020).

Although such adversarial training methods can significantly improve the empirical robustness of neural networks, there is no guarantee that a stronger, newly-discovered attack would not break them. To address this, a separate line of work focuses on certifying robustness against any adversarial perturbations (Raghunathan, Steinhardt, and Liang 2018; Wong and Kolter 2018) but often has difficulty in scaling to large neural networks. Randomized smoothing, a method that injects random additive noises to inputs to construct smoothed classifiers from base classifiers, has been considered the most successful certified defense approach that can be applied to

large neural networks (Lecuyer et al. 2019; Cohen, Rosenfeld, and Kolter 2019; Salman et al. 2019; Yang et al. 2020).

Preemptive image manipulation for robustness There have been a few studies on preemptive image manipulation to protect images from being exploited, yet most of them utilize it for privacy protection. In the facial recognition task, prior work proposes an algorithm that slightly modifies personal photos before uploading them to social media to make them hard to be identified by malicious person recognition systems (Oh, Fritz, and Schiele 2017; Shan et al. 2020; Cherepanova et al. 2021). Our work differs from this prior work in that our goal is to robustify images to be correctly identified by classification systems even under adversarial perturbations.

The most relevant to our work is an approach of Salman et al. (2020), which develops patches that can boost robustness to common corruptions when applied to clean images. However, we consider the problem of manipulating images to be correctly classified against worst-case adversarial perturbations, which are artificially designed to cause misclassification. Ensuring robustness to adversarial perturbations is much more challenging than robustifying images against common corruptions, which are not the worst-case perturbations.

3 Methods

3.1 Problem Setup

To start, we introduce our defense framework for image classification models, along with the adversarial threat model. In our framework, a defender can preemptively modify the original image x_o to produce a new image x_r that is visually indistinguishable from x_o ahead of adversarial attacks. After the modification, the defender discards the original image x_o , so that adversary can only see the modified image x_r . Under this framework, we can consider two types of adversaries:

- A *grey-box* adversary who has complete knowledge of the classification model but does not know the modification algorithm exists.
- A *white-box* adversary who not only has full access to the classification model but also is aware of the existence of the modification algorithm and how it works.

The grey-box adversary will regard the given image x_r as the original image and attempt to find an adversarial example near x_r , as from the conventional adversarial literature. In contrast, the white-box adversary recognizes that x_r is a modified version. Thus, the white-box adversary will instead try to guess the location of the original image x_o and craft an adversarial example near it.

In this paper, we investigate the defender’s optimal strategy for manipulating original images to be resistant against these two adversaries. First, we develop an algorithm that preemptively robustifies original images against the grey-box adversary. Then, we demonstrate our proposed algorithm also exhibits high robustness against adaptively designed white-box attacks.

3.2 Preemptive Robustness

We now formally introduce the concept of *preemptive robustness*. We begin by recalling the definition of adversar-

ial examples. Let $c : \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier which maps images to class labels. Given an original image $x_o \in \mathcal{X}$ and its class $y_o \in \mathcal{Y}$, suppose x_o is correctly classified. Then, an adversarial example $x_o^a \in \mathcal{X}$ of x_o is defined as an image in the neighborhood of x_o such that the classifier changes its prediction, i.e., $c(x_o^a) \neq c(x_o)$ and $x_o^a \in B_\epsilon(x_o)$. Here, $\epsilon > 0$ is the perturbation budget of the adversary and $B_\epsilon(x) = \{x' \in \mathcal{X} \mid \|x' - x\|_p \leq \epsilon\}$ denotes the closed ℓ_p -ball of radius ϵ centered at x . Throughout this paper, we consider $p \in \{2, \infty\}$, the most common settings in adversarial machine learning literature. If the classifier gives robust predictions in the neighborhood of x_o , then we say x_o is robust against adversarial perturbations.

We can extend this notion of adversarial robustness to the whole image space $\mathcal{X}$. To do this, we define the *robust region* of a classifier c as the set of images that c can output robust predictions in the presence of adversarial perturbations.

Definition 1 (ϵ -robust region). *Let $c : \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier and $\epsilon > 0$ be the perturbation budget of an adversary. The ϵ -robust region of the classifier c is defined by $R_\epsilon(c) := \{x \in \mathcal{X} \mid c(x') = c(x), \forall x' \in B_\epsilon(x)\}$.*

Now, consider a defender who can preemptively manipulate x_o under a small modification budget $\delta > 0$ to generate a new image $x_r \in B_\delta(x_o)$, and a grey-box adversary who aims to find an adversarial example near x_r . Then, the defender's optimal strategy against the adversary is to make x_r be correctly classified as y_o and locate in the robust region $R_\epsilon(c)$ so that x_r is robust to adversarial perturbations. If both of these two conditions are satisfied, we say x_o is *preemptively robust* against the grey-box adversary and x_r is a *preemptively robustified image* of x_o .

Definition 2 (Preemptive robustness, grey-box). *Let $c : \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier and $\delta, \epsilon > 0$ be the modification budgets of the defender and the grey-box adversary, respectively. An original image x_o with its class y_o is preemptively robust against the grey-box adversary if there exists $x_r \in B_\delta(x_o)$ such that (i) $c(x_r) = y_o$ and (ii) $x_r \in R_\epsilon(c)$.*

Next, we consider a white-box adversary against the defender. Let us denote the manipulation algorithm of the defender by $m : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{X}$, i.e., $x_r = m(x_o, y_o)$. Then, the white-box adversary will adaptively design its attack algorithm $a_m : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{X}$, which takes x_r and y_o as inputs and produces a candidate of the adversarial example. For the output $a_m(x_r, y_o)$ to be a valid adversarial example, it should be misclassified and located in $B_\epsilon(x_o)$. If the output is not a valid adversarial example, we say x_o is *preemptively robust* against the white-box adversary.

Definition 3 (Preemptive robustness, white-box). *Let $m : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{X}$ be the defender's manipulation algorithm and $a_m : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{X}$ be the adaptive attack algorithm of the white-box adversary. Given an original image x_o and its class y_o , let $x_r = m(x_o, y_o)$ denote the resulting image of the defender's algorithm. Then, x_o is called preemptively robust against the white-box adversary if either of the following conditions is satisfied: (i) $c(a_m(x_r, y_o)) = y_o$ or (ii) $a_m(x_r, y_o) \notin B_\epsilon(x_o)$.*

Note that since the white-box adversary does not have any information about the original x_o in the MitM setting,

forcing $a_m(x_r, y_o)$ to lie in $B_\epsilon(x_o)$ is a non-trivial task for the adversary.

3.3 Preemptive Robustification Algorithm

In this subsection, we develop an algorithm for preemptively robustifying original images against the grey-box adversary. Given a classifier c , finding a preemptively robustified image x_r from an original image x_o can be formulated as the following optimization problem, which is directly from Definition 2:

$$\begin{aligned} & \underset{x_r}{\text{minimize}} \quad \mathbb{1}_{c(x_r) \neq y_o} + \mathbb{1}_{x_r \notin R_\epsilon(c)} \\ & \text{subject to} \quad \|x_r - x_o\|_p \leq \delta, \end{aligned}$$

where $\mathbb{1}$ is the 0-1 loss function.

Note that in this formulation, the defender requires the ground-truth label y_o . However, images in real-world applications are usually unlabeled unless users manually annotate their images. Therefore, it is natural to assume that the defender does not have access to the ground-truth label y_o . In this case, we utilize the classifier's prediction $c(x_o)$ instead of y_o :

$$\begin{aligned} & \underset{x_r}{\text{minimize}} \quad \mathbb{1}_{c(x_r) \neq c(x_o)} + \mathbb{1}_{x_r \notin R_\epsilon(c)} \\ & \text{subject to} \quad \|x_r - x_o\|_p \leq \delta. \end{aligned}$$

As $x_r \notin R_\epsilon(c)$ implies there exists an adversarial example $x_r^a \in B_\epsilon(x_r)$ such that $c(x_r^a) \neq c(x_r)$, we can reformulate the optimization problem as

$$\begin{aligned} & \underset{x_r}{\text{minimize}} \quad \mathbb{1}_{c(x_r) \neq c(x_o)} + \sup_{x_r^a} \mathbb{1}_{c(x_r^a) \neq c(x_r)} \\ & \text{subject to} \quad \|x_r - x_o\|_p \leq \delta \text{ and } \|x_r^a - x_r\|_p \leq \epsilon. \end{aligned}$$

Since 0-1 loss is not differentiable, we use the cross-entropy loss $\ell : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ of the classifier c as the convex surrogate loss function:

$$\begin{aligned} & \underset{x_r}{\text{minimize}} \quad \ell(x_r, c(x_o)) + \sup_{x_r^a} \ell(x_r^a, c(x_r)) \\ & \text{subject to} \quad \|x_r - x_o\|_p \leq \delta \text{ and } \|x_r^a - x_r\|_p \leq \epsilon. \end{aligned} \quad (1)$$

Let $h(x_r)$ denote the objective in Equation (1). Instead of minimizing $h(x_r)$ directly, we minimize $\tilde{h}(x_r) = \sup_{x_r^a \in B_\epsilon(x_r)} \ell(x_r^a, c(x_o))$ since it upper bounds $h(x_r)$ when sufficiently minimized due to Lemma 1.

Lemma 1. *If $\tilde{h}(x_r) \leq -\log(0.5) \simeq 0.6931$, then $h(x_r) \leq 2\tilde{h}(x_r)$.*

Proof. See Supplementary A.1. $\square$

Finally, we have the following optimization problem:

$$\begin{aligned} & \underset{x_r}{\text{minimize}} \quad \sup_{x_r^a} \ell(x_r^a, c(x_o)) \\ & \text{subject to} \quad \|x_r - x_o\|_p \leq \delta \text{ and } \|x_r^a - x_r\|_p \leq \epsilon. \end{aligned} \quad (2)$$

To solve Equation (2), we first approximate the inner maximization problem by running T -step PGD (Madry et al. 2017) whose dynamics is given by

$$\begin{aligned} x_r^{a,0} &= x_r + \eta && \text{(random start)} \\ \tilde{x}_r^{a,t} &= f(x_r^{a,t-1}; c(x_o), \ell) && \text{(adversarial update)} \\ x_r^{a,t} &= \Pi_{x_r, \epsilon}(\tilde{x}_r^{a,t}), && \text{(projection)} \end{aligned}$$

Algorithm 1: Preemptive robustification algorithm

input Original image and its prediction $(x_o, c(x_o))$
 $x_r = x_o$ // or randomly initialized in $B_\delta(x_o)$
for $i = 1, \dots, \text{MAXITER}$ **do**
 // Generate N PGD adversarial examples
 for $n = 1, \dots, N$ **do**
 $x_{r,n}^a = x_r + \eta$ where $\eta \sim \mathcal{U}(B_\epsilon(0))$
 for $t = 1, \dots, T$ **do**
 $x_{r,n}^a \leftarrow \Pi_{x_r, \epsilon}(f(x_{r,n}^a; c(x_o), \ell))$
 end for
 end for
 // Update image
 $x_r \leftarrow \Pi_{x_o, \delta}\left(x_r - \beta \cdot \frac{1}{N} \sum_{n=1}^N \frac{\partial \ell(x_{r,n}^a, c(x_o))}{\partial x_r}\right)$
end for
output x_r

where η is a noise uniformly sampled from $B_\epsilon(0)$, f is FGSM (Goodfellow, Shlens, and Szegedy 2015) defined as

$$f(x; y, \ell) = \begin{cases} x + \alpha \cdot \text{sgn}(\nabla_x \ell(x, y)) & \text{if } p = \infty \\ x + \alpha \cdot \frac{\nabla_x \ell(x, y)}{\|\nabla_x \ell(x, y)\|_2} & \text{if } p = 2, \end{cases}$$

and $\Pi_{x_r, \epsilon}$ is a projection operation onto $B_\epsilon(x_r)$. Then, we iteratively solve the approximate problem given by replacing x_r^a to $x_r^{a,T}$ in Equation (2). To update x_r , we compute the gradient of $\ell(x_r^a, c(x_o))$ with respect to x_r expressed as

$$\frac{\partial \ell(x_r^a, c(x_o))}{\partial x_r} = \nabla_x \tilde{f}(x_r^{a,0})^\top \cdot \dots \cdot \nabla_x \tilde{f}(x_r^{a,T-1})^\top \cdot \nabla_x \ell(x_r^{a,T}, c(x_o)),$$

where $\nabla_x \tilde{f}$ is the Jacobian matrix of $\tilde{f} = \Pi_{x_r, \epsilon} \circ f$ which can be computed via back-propagation. After computing the gradient, we update x_r by projected gradient descent method:

$$x_r \leftarrow \Pi_{x_o, \delta}\left(x_r - \beta \cdot \frac{\partial \ell(x_r^a, c(x_o))}{\partial x_r}\right).$$

Note that $\ell(x_r^a, c(x_o))$ is a random variable dependent on η . Therefore, we generate N adversarial examples $\{x_{r,n}^a\}_{n=1}^N$ with different noises and optimize the sample mean of the losses instead. Algorithm 1 shows the overall preemptive robustification algorithm and Figure 2 illustrates the optimization process. Some examples of robustified images generated from our algorithm are shown in Supplementary D.

3.4 Computing Update Gradient without Second-Order Derivatives

Computing the update gradient with respect to x_r involves the use of second-order derivatives of the loss function ℓ since the dynamics f contains the loss gradient $\nabla_x \ell$. Standard deep learning libraries, such as PyTorch (Paszke et al. 2019), support the computation of these higher-order derivatives. However, it imposes a huge memory burden as the size of the computational graph increases. Furthermore, when $p = 2$, computing the update gradient with the second-order

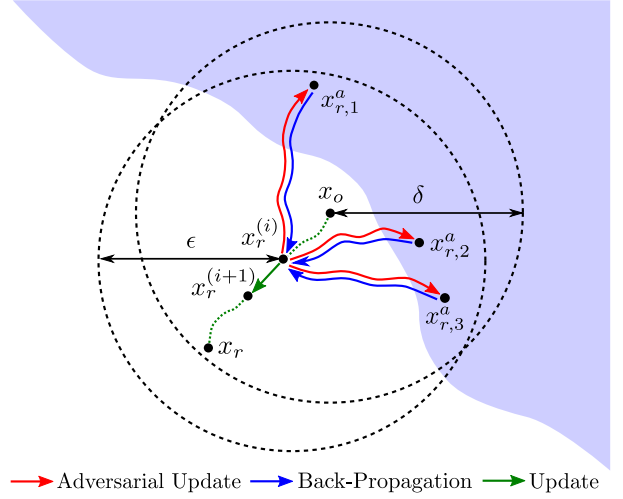


Figure 2: Illustration of the preemptive robustification process. The shaded region represents the set of misclassified points.

derivatives might cause an *exploding gradient problem* if the loss gradient vanishes by Lemma 2 and Proposition 1.

Lemma 2. Suppose ℓ is twice-differentiable and its second partial derivatives are continuous. If $p = 2$, the Jacobian of the dynamics f is

$$\nabla_x f = I + \alpha \cdot \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right) \frac{H}{\|g\|_2},$$

where $g = \nabla_x \ell$ and $H = \nabla_x^2 \ell$.

Proof. See Supplementary B.1. $\square$

Proposition 1. If the maximum eigenvalue of H in absolute value is σ , then

$$\|\nabla_x \tilde{f}^\top \cdot a\|_2 \leq \left(1 + \alpha \cdot \frac{\sigma}{\|g\|_2} \right) \|a\|_2.$$

Proof. See Supplementary B.2. $\square$

As we update x_r , the loss gradients g of x_r and its intermediate adversarial examples $x_r^{a,t}$ reduce to zero, which might cause the update gradient to explode and destabilize the update process. To address this problem, we approximate the update gradient by excluding the second-order derivatives, following the practice in Finn, Abbeel, and Levine (2017). We also include an experiment in comparison to using the exact update gradient in Supplementary B.3. For the case of $p = \infty$, the second-order derivatives naturally vanish since we take the sign of the loss gradient $\nabla_x \ell(x, y)$. Therefore, the approximate gradient is equal to the exact update gradient.

3.5 Network Training Scheme for Improving Preemptive Robustness

So far, we have explored how the defender can preemptively robustify the original image, given a pre-trained classifier. Now, we explore the defender's network training scheme for

a classifier where data points are preemptively robust with high probability. Suppose the defender has a labeled training set, which is drawn from a true data distribution $\mathcal{D}$. To induce data points to be preemptively robust, the defender's optimal training objective should have the following form:

$$\begin{aligned} & \underset{\theta}{\text{minimize}} \quad \mathbb{E}_{(x_o, y_o) \sim \mathcal{D}} [\ell(\hat{x}_r^a, y_o; \theta)] \\ & \text{subject to} \quad \hat{x}_r^a = \underset{x_r^a \in B_\epsilon(\hat{x}_r)}{\text{argmax}} \ell(x_r^a, y_o) \\ & \quad \quad \quad \hat{x}_r = \underset{x_r \in B_\delta(x_o)}{\text{argmin}} \sup_{x_r^a \in B_\epsilon(x_r)} \ell(x_r^a, y_o), \end{aligned}$$

where θ is the set of trainable parameters. Concretely, the defender first attempts to craft a candidate for preemptively robustified points $\hat{x}_r$ of the original data point x_o . Then, the defender generates an adversarial example $\hat{x}_r^a$ of $\hat{x}_r$ and minimizes its cross-entropy loss $\ell(\hat{x}_r^a, y_o; \theta)$ so that $\hat{x}_r$ becomes resistant to adversarial perturbations. Note that the ground-truth label y_o is used instead of the prediction $c(x_o)$, since we assume the ground-truth label of the training set is given.

The most direct way to optimize the objective would be to find $\hat{x}_r$ from x_o using our preemptive robustification algorithm and perform K -step PGD adversarial training (Madry et al. 2017) with $\hat{x}_r$. However, since our algorithm requires running T -step PGD dynamics per each update, the proposed training procedure would be more computationally demanding than standard PGD adversarial training. To ease this problem, we replace the inner maximization $\sup_{x_r^a} \ell(x_r^a, y_o)$ in the preemptive robustification process by $\ell(x_r, y_o)$:

$$\begin{aligned} \hat{x}_r &= \underset{x_r \in B_\delta(x_o)}{\text{argmin}} \sup_{x_r^a \in B_\epsilon(x_r)} \ell(x_r^a, y_o) \\ \implies \hat{x}_r &= \underset{x_r \in B_\delta(x_o)}{\text{argmin}} \ell(x_r, y_o). \end{aligned}$$

Then, $\hat{x}_r$ can be easily computed by running L -step PGD on x_o towards minimizing the cross-entropy loss. We denote this training scheme as *preemptively robust training*. The full training procedure is summarized in Algorithm 2 (differences with the standard adversarial training marked in blue).

Note that the standard adversarial training is a specific case of our preemptively robust training, forcing training data x_o to be far from the decision boundary. However, recent work demonstrates there is a trade-off between the classification error and the boundary error, which is why standard adversarial training significantly decreases the clean accuracy (Tsipras et al. 2019; Zhang et al. 2019b). In contrast, our proposed training scheme allows original images x_o to lie near the decision boundary and only enforces the preemptively robustified images $\hat{x}_r$ to be distant from the boundary. Our experiments in Section 4 show preemptively robust training is less prone to suffer from the clean accuracy drop due to this flexibility.

3.6 Preemptive Robustification for Classifiers with Randomized Smoothing

Our preemptive robustification algorithm can also be applied to smoothed classifiers. Given a base classifier $c : \mathcal{X} \rightarrow \mathcal{Y}$, a smoothed classifier $\tilde{c} : \mathcal{X} \rightarrow \mathcal{Y}$ is defined as

$$\tilde{c}(x) = \underset{y \in \mathcal{Y}}{\text{argmax}} \mathbb{P}(c(x + \xi) = y),$$

Algorithm 2: Preemptively robust training, $p = \infty$

```

input Training dataset  $\mathcal{D}_{train}$ , maximum epoch  $N$ 
for  $n = 1, \dots, N$  do
  for  $(x_o, y_o) \in \mathcal{D}_{train}$  do
    // Do  $L$ -step PGD towards minimizing loss
     $x_r = x_o + \eta$  where  $\eta \sim B_\delta(0)$ 
    for  $l = 1, \dots, L$  do
       $x_r \leftarrow \Pi_{x_o, \delta}(x_r - \beta \cdot \text{sgn}(\nabla_x \ell(x_r, y_o)))$ 
    end for
    // Do  $K$ -step PGD towards maximizing loss
     $x_r^a = x_r + \eta$  where  $\eta \sim B_\epsilon(0)$ 
    for  $t = 1, \dots, K$  do
       $x_r^a \leftarrow \Pi_{x_r, \epsilon}(x_r^a + \alpha \cdot \text{sgn}(\nabla_x \ell(x_r^a, y_o)))$ 
    end for
     $\theta \leftarrow \theta - \nabla_\theta \ell(x_r^a, y_o)$ 
  end for
end for

```

where $\xi \sim \mathcal{N}(0, \sigma^2 I)$. Crafting adversarial examples x_r^a for the smoothed classifier, which is necessary for approximating the inner maximization in Equation (2), is ill-behaved since the argmax operation is non-differentiable. To address this problem, we follow the practice in Salman et al. (2019) and approximate the smoothed classifier $\tilde{c}$ with the smoothed soft classifier $\tilde{C} : \mathcal{X} \rightarrow P(\mathcal{Y})$ defined as

$$\tilde{C}(x) = \mathbb{E}_{\xi \sim \mathcal{N}(0, \sigma^2 I)} [C(x + \xi)], \quad (3)$$

where $P(\mathcal{Y})$ is the set of probability distribution over $\mathcal{Y}$ and $C : \mathcal{X} \rightarrow P(\mathcal{Y})$ is the soft version of the base classifier c such that $\text{argmax}_{y \in \mathcal{Y}} C(x)_y = c(x)$. Finally, the adversarial example x_r^a can be found by maximizing the cross-entropy loss of $\tilde{C}$ instead:

$$\begin{aligned} & \underset{x_r^a}{\text{maximize}} \quad -\log(\tilde{C}(x_r^a)_{c(x_o)}) \\ & \text{subject to} \quad \|x_r^a - x_r\|_p \leq \epsilon, \end{aligned}$$

which can be approximated by T -step randomized PGD (Salman et al. 2019), where ξ is sampled M times to compute the sample mean of Equation (3) at each step. By replacing the inner maximization problem in Equation (2) with the randomized PGD, we can update x_r in a similar process.

3.7 Adaptive Attack against Preemptive Robustification

So far, we have developed preemptive defense strategies against the grey-box adversary. Now, we consider the white-box adversary described in Section 3.1, which is aware that the given image x_r has been preemptively modified and aims to craft an adversarial example near the original image x_o that is unknown. The most direct way for the adversary to achieve this is to reconstruct x_o from x_r and apply standard attack algorithms (e.g., PGD) on the reconstructed image $\hat{x}_o$. Since we assume the adversary knows the detailed hyperparameter settings of the robustification algorithm, the adversary can leverage this information to approximate the inverse dynamics of the preemptive robustification process starting from x_r ,

Algorithm 3: Original image reconstruction

input Preemptively robustified image and its prediction
 $(x_r, c(x_r))$
 $\hat{x}_o = x_r$
for $i = 1, \dots, \text{MAXITER}$ **do**
 // Generate N adversarial examples
 for $n = 1, \dots, N$ **do**
 $\hat{x}_{o,n}^a = \hat{x}_o + \eta$ where $\eta \sim \mathcal{U}(B_\epsilon(0))$
 for $t = 1, \dots, T$ **do**
 $\hat{x}_{o,n}^a \leftarrow \Pi_{\hat{x}_o, \epsilon}(f(\hat{x}_{o,n}^a; c(x_r), \ell))$
 end for
 end for
 // Update image
 $\hat{x}_o \leftarrow \Pi_{x_r, \delta}\left(\hat{x}_o + \beta \cdot \frac{1}{N} \sum_{n=1}^N \frac{\partial \ell(\hat{x}_{o,n}^a, c(x_r))}{\partial \hat{x}_o}\right)$
end for
output $\hat{x}_o$

Algorithm 4: Adaptive white-box attack

input Preemptively robustified image and its prediction
 $(x_r, c(x_r))$, target y_o
 // Reconstruct original image
 $\hat{x}_o = \text{ORIGINALIMAGERECONSTRUCTION}(x_r, c(x_r))$
 // Run standard attack algorithm on reconstructed image
 $\hat{x}_o^a = \text{ATTACKALGORITHM}(\hat{x}_o, y_o, \epsilon')$
output $\hat{x}_o^a$

as described in Algorithm 3. The only difference between this reconstruction algorithm and the preemptive robustification process is the initialization and the update direction, modified to suit the reconstruction objective. Algorithm 4 shows the overall procedure of the adaptive white-box attack. Note that the adversary may modify $\hat{x}_o$ with a budget smaller than ϵ to induce $\hat{x}_o^a \in B_\epsilon(x_o)$, considering the original image might not be reconstructed accurately.

Figure 3 shows the proposed reconstruction algorithm performs well in terms of reconstruction error if the preemptive robustification algorithm starts from the original image itself. However, as we run the preemptive robustification algorithm starting from a random point in $B_\delta(x_o)$, the performance of the reconstruction algorithm degrades considerably. About 80% of the reconstructed images locate near the boundaries of ϵ -balls centered at original images, which shows the difficulty of reconstructing the original image. We also observe that most of the resulting white-box attack examples generated from the reconstructed images lie outside $B_\epsilon(x_o)$, which implies they are not valid adversarial examples.

4 Experiments

We evaluate our methods on CIFAR-10 and ImageNet by measuring classification accuracies of preemptively robustified images under the grey-box and white-box adversaries. As it is natural to assume that the defender and the adversary have the same modification budget, we set $\delta = \epsilon$ for all experiments. Both the adversaries use 20-step untargeted PGD and AutoAttack (Croce and Hein 2020) to find adversarial

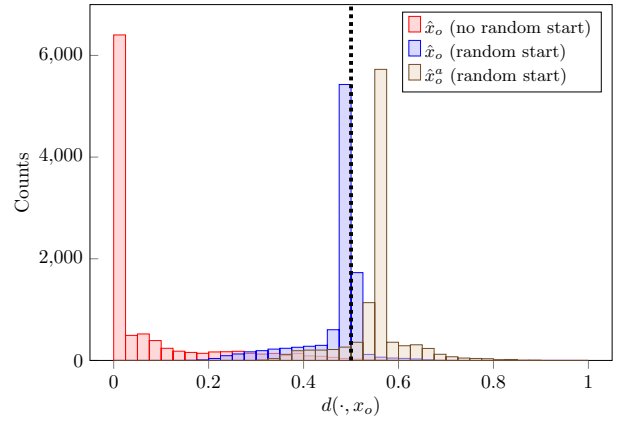


Figure 3: Histograms of the distances between original images x_o and the reconstructed images $\hat{x}_o$ or their white-box attack examples $\hat{x}_o^a$ on CIFAR-10 test set. We use a preemptively robust model with $p = 2$ and $\delta = \epsilon = 0.5$. The dotted line indicates the adversary’s perturbation bound ϵ .

examples. For the white-box adversary, we sweep the final perturbation budget ϵ' and report the lowest accuracy measured. More details are listed in Supplementary C. The code is available online ¹.

4.1 CIFAR-10

We consider two types of perturbations: ℓ_∞ with $\epsilon = 8/255$ and ℓ_2 with $\epsilon = 0.5$. To show the effectiveness of our robustification algorithm, we also report results without preemptive robustification (None)². As the baseline for our preemptively robust model, we use an adversarially trained model with early stopping (Rice, Wong, and Kolter 2020) (ADV).

The results in Table 1 and Table 2 show that our preemptive manipulation method successfully robustifies images compared to without the manipulation, achieving adversarial accuracies higher than 80% in both the perturbation settings. Also, while the adversarially trained model does induce some images to be preemptively robust, our proposed network training method further boosts the performance of the robustified images in both the clean and adversarial accuracies.

Note that the white-box attacks are not effective on ℓ_∞ as they fail to lay their adversarial examples within the $B_\epsilon(x_o)$ ball. This is in part due to the characteristic of the ℓ_∞ distance measure, as ℓ_∞ distance can spike even when a single pixel deviates from the original value. On the other hand, white-box attacks on ℓ_2 perturbations succeed to pose reasonable threats. However, the worst-case adversarial accuracy is still over 10% higher than without our methods.

4.2 ImageNet

We consider two types of perturbations: ℓ_∞ with $\epsilon = 4/255$ and ℓ_2 with $\epsilon = 3.0$. As with the CIFAR-10 experiments, we compare our preemptive robustification algorithm to without

¹https://github.com/snu-mlab/preemptive_robustification

²In this case, the grey-box adversary is the same as the white-box adversary since no modification occurs on original images.

Model	Preempt.	Clean	Grey-box		White-box	
			PGD	AA	PGD	AA
ADV	None	86.72	54.59	51.68	54.59	51.68
ADV	Ours	86.72	86.23	81.70	86.72	86.72
Ours	Ours	88.54	87.10	82.88	88.54	88.54

Table 1: CIFAR-10 classification accuracy under grey-box and white-box adversaries with ℓ_∞ perturbation, $\epsilon = 8/255$.

Model	Preempt.	Clean	Grey-box		White-box	
			PGD	AA	PGD	AA
ADV	None	90.85	71.90	71.21	71.90	71.21
ADV	Ours	90.85	84.81	83.56	85.12	79.48
Ours	Ours	92.57	91.81	89.32	85.02	80.79

Table 2: CIFAR-10 classification accuracy under grey-box and white-box adversaries with ℓ_2 perturbation, $\epsilon = 0.5$.

the algorithm (None). As the baseline, we use a model adversarially trained with the fast training schemes (Wong, Rice, and Kolter 2020) for computational efficiency (ADV).

Table 3 and Table 4 show our preemptive robustification methods scale to more practical datasets with bigger images. Our methods allow the natural images to be much more robust to adversarial attacks, with over 15% higher worst-case adversarial accuracies, and at the same time maintains higher clean accuracies. We observe that similar to the CIFAR-10 experiments, the white-box attacks on the ℓ_∞ distance measure are not successful in finding appropriate adversarial samples.

Model	Preempt.	Clean	Grey-box		White-box	
			PGD	AA	PGD	AA
ADV	None	56.24	32.03	27.52	32.03	27.52
ADV	Ours	56.24	55.79	47.14	56.24	56.24
Ours	Ours	61.01	59.66	48.24	61.01	61.01

Table 3: ImageNet classification accuracy under grey-box and white-box adversaries with ℓ_∞ perturbation, $\epsilon = 4/255$.

Model	Preempt.	Clean	Grey-box		White-box	
			PGD	AA	PGD	AA
ADV	None	54.99	32.07	27.58	32.07	27.58
ADV	Ours	55.05	51.70	43.32	46.38	37.49
Ours	Ours	61.60	58.13	43.60	54.23	47.54

Table 4: ImageNet classification accuracy under grey-box and white-box adversaries with ℓ_2 perturbation, $\epsilon = 3.0$.

4.3 Randomized Smoothing

We also evaluate our preemptive robustification algorithm for smoothed classifiers. We consider ℓ_2 perturbations, where $\epsilon = 0.5$ for CIFAR-10 and $\epsilon = 3.0$ for ImageNet. We utilize

a smoothed model trained with Gaussian noise augmentation as proposed in Cohen, Rosenfeld, and Kolter (2019) due to its simplicity. We measure empirical adversarial accuracies using 20-step randomized PGD and its 10 restart version (PGD-10). We also compute the certified radii of the images and measure the certified adversarial accuracies.

Table 5 and Table 6 shows the empirical robustness results against the randomized PGD. We observe our methods can significantly enhance preemptive robustness on the smoothed classifiers, maintaining 28% and 49% higher the worst-case adversarial accuracies than the baseline on CIFAR10 and ImageNet, respectively. The results in Table 7 show our method also improves the certified robustness on the smoothed networks. Our methods achieve 22% and 15% higher certified accuracies on CIFAR10 and ImageNet, respectively.

Preempt.	Clean	Grey-box		White-box	
		PGD	PGD-10	PGD	PGD-10
None	92.14	56.02	53.01	56.02	53.01
Ours	92.35	91.37	89.98	82.06	80.71

Table 5: CIFAR-10 empirical accuracy of smoothed network under grey-box and white-box adversaries with ℓ_2 perturbation, $\epsilon = 0.5$. We set the noise level to $\sigma = 0.1$.

Preempt.	Clean	Grey-box		White-box	
		PGD	PGD-10	PGD	PGD-10
None	69.93	9.61	8.61	9.61	8.61
Ours	70.05	62.27	57.24	68.05	67.72

Table 6: ImageNet empirical accuracy of smoothed network under grey-box and white-box adversaries with ℓ_2 perturbation, $\epsilon = 3.0$. We set the noise level to $\sigma = 0.25$.

Preempt.	Clean	Cert.	Preempt.	Clean	Cert.
None	82.84	55.58	None	47.02	12.68
Ours	84.72	77.95	Ours	52.66	27.89

Table 7: CIFAR-10 (left) and ImageNet (right) certified accuracies of smoothed network with ℓ_2 perturbation. The noise levels are $\sigma = 0.25$ for CIFAR-10 and $\sigma = 1.0$ for ImageNet.

5 Conclusion

We consider a real-world adversarial framework where the MitM adversary intercepts and manipulates the images during transmission. To protect users from such attacks, we introduce a novel optimization algorithm for finding robust points in the vicinity of original images along with a new network training method suited for enhancing preemptive robustness. The experiments show that our algorithm can find such robust points for most of the correctly classified images. Further results show our method also improves preemptive robustness on smooth classifiers.

Acknowledgement

This work was supported in part by SNU-NAVER Hyper-scale AI Center and 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 No. 2019-0-01371, Development of brain-inspired AI with human-like intelligence). 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

- Cherepanova, V.; Goldblum, M.; Foley, H.; Duan, S.; Dickerson, J.; Taylor, G.; and Goldstein, T. 2021. LowKey: leveraging adversarial attacks to protect social media users from facial recognition. In *ICLR*.
- Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified adversarial robustness via randomized smoothing. In *ICML*.
- Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*.
- Desmedt, Y. 2011. *Man-in-the-Middle Attack*. Springer.
- Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.
- Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *ICLR*.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.
- Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.
- Lecuyer, M.; Atlidakis, V.; Geambasu, R.; Hsu, D.; and Jana, S. 2019. Certified robustness to adversarial examples with differential privacy. In *IEEE Symposium on Security and Privacy (SP)*.
- Li, S.; Neupane, A.; Paul, S.; Song, C.; Krishnamurthy, S. V.; Roy-Chowdhury, A. K.; and Swami, A. 2019. Adversarial Perturbations Against Real-Time Video Classification Systems. In *NDSS*.
- Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. In *ICLR*.
- Oh, S. J.; Fritz, M.; and Schiele, B. 2017. Adversarial image perturbation for privacy protection a game theory perspective. In *ICCV*.
- Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*.
- Raghunathan, A.; Steinhardt, J.; and Liang, P. 2018. Certified defenses against adversarial examples. In *ICLR*.
- Rice, L.; Wong, E.; and Kolter, J. Z. 2020. Overfitting in adversarially robust deep learning. In *ICML*.
- Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*.
- Salman, H.; Ilyas, A.; Engstrom, L.; Vemprala, S.; Madry, A.; and Kapoor, A. 2020. Unadversarial Examples: Designing Objects for Robust Vision. arXiv:2012.12235.
- Salman, H.; Li, J.; Razenshteyn, I.; Zhang, P.; Zhang, H.; Bubeck, S.; and Yang, G. 2019. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*.
- Santurkar, S.; Tsipras, D.; Tran, B.; Ilyas, A.; Engstrom, L.; and Madry, A. 2019. Comoputer vision with a single (robust) classifier. In *NeurIPS*.
- Shafahi, A.; Najibi, M.; Ghiasi, A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L.; Taylor, G.; and Goldstein, T. 2019. Adversarial Training for Free! In *NeurIPS*.
- Shan, S.; Wenger, E.; Zhang, J.; Li, H.; Zheng, H.; and Zhao, B. Y. 2020. Fawkes: Protecting privacy against unauthorized deep learning models. In *29th USENIX Security Symposium (USENIX Security 20)*.
- Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. In *ICLR*.
- Tieleman, T.; Hinton, G.; et al. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31.
- Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; and Madry, A. 2019. Robustness may be at odds with accuracy. In *ICLR*.
- Wang, D.; Li, C.; Wen, S.; Nepal, S.; and Xiang, Y. 2019. Man-in-the-Middle Attacks against Machine Learning Classifiers via Malicious Generative Models. In *IEEE Transactions on Dependable and Secure Computing*.
- Wong, E.; and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*.
- Wong, E.; Rice, L.; and Kolter, J. Z. 2020. Fast is better than free: Revisiting adversarial training. In *ICLR*.
- Yang, G.; Duan, T.; Hu, J. E.; Salman, H.; Razenshteyn, I.; and Li, J. 2020. Randomized smoothing of all shapes and sizes. In *ICML*.
- Zagoruyko, S.; and Komodakis, N. 2017. Wide Residual Networks. arXiv:1605.07146.
- Zhang, D.; Zhang, T.; Lu, Y.; Zhu, Z.; and Dong, B. 2019a. You only propagate once: Accelerating adversarial training via maximal principle. In *NeurIPS*.
- Zhang, H.; and Wang, J. 2019. Defense against adversarial attacks using feature scattering-based adversarial training. In *NeurIPS*.
- Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; Ghaoui, L. E.; and Jordan, M. I. 2019b. Theoretically Principled Trade-off between Robustness and Accuracy. In *ICML*.
- Zhang, J.; Xu, X.; Han, B.; Niu, G.; Cui, L.; Sugiyama, M.; and Kankanhalli, M. 2020. Attacks Which Do Not Kill Training Make Adversarial Learning Stronger. In *ICML*.

A Preemptive Robustification Algorithm

A.1 Proof of Lemma 1

Since $x_r^a \in B_\epsilon(x_r)$, we have

$$\ell(x_r, c(x_o)) \leq \sup_{x_r^a} \ell(x_r^a, c(x_o)) = \tilde{h}(x) \leq -\log(0.5). \quad (4)$$

Let $C(x_r)$ be the softmax probability of x_r . Equation (4) implies $C(x_r)_{c(x_o)} \geq 0.5$, i.e., $c(x_r) = c(x_o)$. Finally, we have

$$\begin{aligned} h(x_r) &= \ell(x_r, c(x_o)) + \sup_{x_r^a} \ell(x_r^a, c(x_r)) \\ &= \ell(x_r, c(x_o)) + \sup_{x_r^a} \ell(x_r^a, c(x_o)) \\ &\quad (\because c(x_r) = c(x_o)) \\ &\leq 2 \sup_{x_r^a} \ell(x_r^a, c(x_o)) \\ &\quad (\because \ell(x_r, c(x_o)) \leq \sup_{x_r^a} \ell(x_r^a, c(x_o))) \\ &= 2\tilde{h}(x_r). \end{aligned}$$

□

A.2 Satisfiability of the Assumption in Lemma 1

Lemma 1 requires the assumption that the loss $\tilde{h}(x_r) = \sup_{x_r^a \in B_\epsilon(x_r)} \ell(x_r^a, c(x_o))$ should be less or equal to $-\log(0.5)$. Our preemptive robustification algorithm can sufficiently minimize the loss to meet the assumption. To verify this, we measure the loss values of the resulting preemptively robustified images x_r from CIFAR-10 test images that are correctly classified. Figure 4 shows that 99.3% of the images satisfy the assumption.

B Computing Update Gradient without Second-Order Derivatives

B.1 Proof of Lemma 2

It is enough to compute the Jacobian of $\frac{g}{\|g\|_2}$. By the quotient rule, we have

$$\begin{aligned} \nabla_x \left(\frac{g}{\|g\|_2} \right) &= \frac{\|g\|_2 \cdot H - g \cdot (\nabla_x \|g\|_2)^\top}{\|g\|_2^2} \\ &= \frac{H}{\|g\|_2} - \frac{g \cdot (\nabla_x \|g\|_2)^\top}{\|g\|_2^2}. \end{aligned} \quad (5)$$

Now, we compute $\nabla_x \|g\|_2$. Since $\|g\|_2^2 = \langle g, g \rangle$, we have

$$2\|g\|_2 \cdot \nabla_x \|g\|_2 = \nabla_x \langle g, g \rangle = 2H \cdot g. \quad (6)$$

Plugging Equation (6) into Equation (5), we have

$$\begin{aligned} \nabla_x \left(\frac{g}{\|g\|_2} \right) &= \frac{H}{\|g\|_2} - \frac{g \cdot (\nabla_x \|g\|_2)^\top}{\|g\|_2^2} \\ &= \frac{H}{\|g\|_2} - \frac{g \cdot g^\top \cdot H^\top}{\|g\|_2^3} \\ &= \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right) \frac{H}{\|g\|_2}, \end{aligned}$$

since H is symmetric, and this completes the proof. □

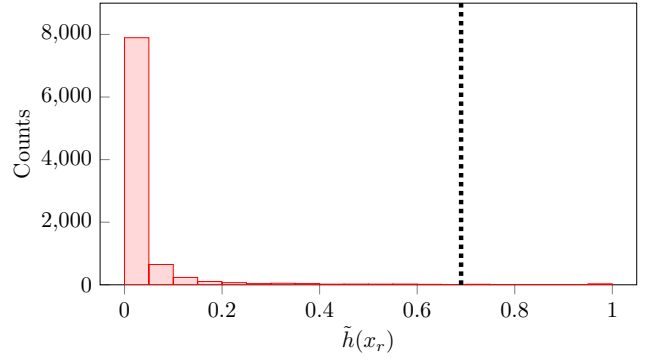


Figure 4: Histogram of the loss values $\tilde{h}(x_r)$ on CIFAR-10 test images. The dotted line indicate the upper bound for $\tilde{h}(x_r)$ to satisfy the assumption in Lemma 1.

B.2 Proof of Proposition 1

Note that $P = I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top$ is a projection map onto a hyperplane whose normal vector is $\frac{g}{\|g\|_2}$. Since a projection map is a contraction map, we have

$$\begin{aligned} \|\nabla_x f^\top a\|_2 &= \left\| \left(I + \alpha \cdot \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right) \frac{H}{\|g\|_2} \right)^\top a \right\|_2 \\ &\quad (\because \text{By Lemma 2}) \\ &= \left\| a + \alpha \cdot \frac{H^\top}{\|g\|_2} \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right)^\top a \right\|_2 \\ &= \|a\|_2 + \left\| \alpha \cdot \frac{H^\top}{\|g\|_2} \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right)^\top a \right\|_2 \\ &\quad (\because \text{By the triangular inequality}) \\ &\leq \|a\|_2 + \alpha \cdot \frac{\|H^\top\|_2}{\|g\|_2} \cdot \left\| \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right)^\top \right\|_2 \|a\|_2 \\ &\quad (\because \|AB\|_2 \leq \|A\|_2 \|B\|_2) \\ &= \|a\|_2 + \alpha \cdot \frac{\sigma}{\|g\|_2} \cdot \left\| \left(I - \left(\frac{g}{\|g\|_2} \right) \left(\frac{g}{\|g\|_2} \right)^\top \right)^\top \right\|_2 \|a\|_2 \\ &\quad (\because H \text{ is symmetric}) \\ &= \|a\|_2 + \alpha \cdot \frac{\sigma}{\|g\|_2} \cdot \|a\|_2, \quad (\because P \text{ is a contraction map}) \end{aligned}$$

Finally, the projection operator $\Pi_{x_r, \epsilon}$ has a form of $\Pi_{x_r, \epsilon}(x) = k(x - x_r) + x_r$, where $0 < k \leq 1$. Therefore, we have

$$\begin{aligned} \|\nabla_x \tilde{f}^\top a\|_2 &= \|\nabla_x f^\top \nabla_x (\Pi_{x_r, \epsilon})^\top a\|_2 \\ &\quad (\because \tilde{f} = \Pi_{x_r, \epsilon} \circ f) \\ &= k \|\nabla_x f^\top a\|_2 \quad (\because \nabla_x \Pi_{x_r, \epsilon} = kI) \\ &\leq \|\nabla_x f^\top a\|_2 \quad (\because 0 < k \leq 1) \\ &\leq \|a\|_2 + \alpha \cdot \frac{\sigma}{\|g\|_2} \cdot \|a\|_2, \end{aligned}$$

which completes the proof. □

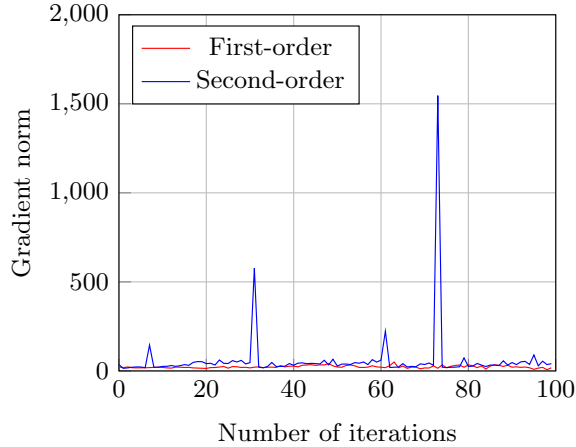


Figure 5: The ℓ_2 norm of update gradient at each iteration during preemptive robustification process with the first-order approximation and with the exact computation using second-order derivatives for an image in CIFAR-10 test set. The approximate gradient update can successfully robustify the image, while the exact gradient update does not.

B.3 Experiments with Second-Order Derivatives

To examine whether the exact gradient computation incurs exploding update gradients, we measure the ℓ_2 norm of the update gradient computed with second-order derivatives at each step, compared to its first-order approximation. We run our experiment on a CIFAR-10 classifier and consider ℓ_2 perturbations under $\delta = \epsilon = 0.5$. Figure 5 shows that preemptive robustification with exact gradient computation leads to unstable update gradients, in contrast to the approximate gradient update.

C Implementation Details

We fix $\text{MAXITER} = 100$, $T = 20$, $\alpha = \epsilon/4$, $N = 1$, and $L = 1$ throughout all the experiments. For implementing the preemptive robustification algorithm, we transform $B_\delta(x_o)$ to $\mathbb{R}^n$ via $\tanh$ transformation and use RMSProp optimizer (Tieleman, Hinton et al. 2012) to update preemptively robustified images when dealing with ℓ_∞ perturbations. Under ℓ_2 perturbations, we use projected gradient descent method as default. For implementing the adaptive white-box attack, we sweep the final perturbation budget of the adversary with $\epsilon' \in \{0.25\epsilon, 0.5\epsilon, 0.75\epsilon, 1.0\epsilon\}$.

C.1 Experiments on CIFAR-10 and ImageNet

CIFAR-10 For network training, we use Wide ResNet architecture (Zagoruyko and Komodakis 2017) with depth 34 and width 10. We train the models for 200 epochs with an initial learning rate of 0.1, decayed with a factor of 0.1 on epoch 100 and 150. We use SGD optimizer with weight decay $5E - 4$, momentum 0.9, and batch size 128. We trained ADV models (Rice, Wong, and Kolter 2020) using 10-step PGD with a step size of $\epsilon/4$ and select the checkpoint with the best performance on the test set. For preemptively robust

models, we use the same 10-step PGD for adversarial example generations, set β during model training to ϵ , and choose the latest checkpoint.

We take the whole 10,000 images from CIFAR-10 test set and run our preemptive robustification algorithm on them. On ℓ_∞ perturbations with $\epsilon = 8/255$, we set the learning rate $\beta = 0.1$, and on ℓ_2 perturbations with $\epsilon = 0.5$, we set $\beta = 0.001$. Each β is chosen by tuning between a range of $\{1.0, 0.5, 0.1, 0.05, 0.01, 0.001\}$ and $\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$.

ImageNet For network training, we use ResNet-50 architecture (He et al. 2016). For ADV models, we train the models with adversarial training using 2-step PGD to reduce the training cost. The PGD step size is set to ϵ , following the protocol of Wong, Rice, and Kolter (2020). Additionally, we adopt the learning rate and input preprocessing schedule of Wong, Rice, and Kolter (2020), except that we set the input image size to 224×224 in phase 3, instead of 288×288 . For preemptively robust models, we use the 2-step PGD for adversarial example generation and tune δ and β during model training to $\epsilon/2$.

We randomly sample 10,000 images from the test set, randomly crop and resize them to 224×224 , and run our preemptive robustification algorithm on the transformed images. We consider ℓ_∞ perturbations with $\epsilon = 4/255$ and ℓ_2 perturbations with $\epsilon = 3.0$. On the ℓ_∞ perturbations, β is set to 0.1, chosen by tuning between a range of $\{1.0, 0.5, 0.1, 0.05, 0.01\}$. On ℓ_2 perturbations, we tuned β between a range of $\{1.0, 0.1, 0.01, 0.001, 0.0001\}$ and choose 0.1.

C.2 Experiments on Randomized Smoothing

We use the same network structures from Section 4.1 and 4.2 and train networks with additive Gaussian noises drawn from $\mathcal{N}(0, \sigma^2 I)$. For smoothed models on ImageNet, we use the pre-trained models provided by Cohen, Rosenfeld, and Kolter (2019). We take 50 Gaussian samples and choose the most probable class by a majority vote for class prediction.

To evaluate the empirical robustness on the smoothed classifiers, we set the noise level σ to 0.1 on CIFAR-10 and 0.25 on ImageNet. The PGD settings are the same as in the experiments for base classifiers. We take 5 Gaussian samples for preemptive robustification and 50 Gaussian samples for evaluation.

To measure the certified robustness on the smoothed classifiers, we set noise levels to be larger than those used in evaluating the empirical robustness since a larger σ leads to larger certified radii. We set σ to 0.25 and 1.0 for the smoothed classifiers on CIFAR-10 and ImageNet, respectively. We take 100,000 Gaussian samples to compute certified radii of images.

For CIFAR-10, the learning rate β is set to 0.005, chosen within a range of $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$. For ImageNet, the learning rate β is set to 0.1, chosen within a range of $\{1.0, 0.5, 0.1, 0.05, 0.01\}$.

C.3 Computing Infrastructure

We conduct all experiments on Ubuntu 16.04 with a hardware system comprised of Intel Xeon Gold 5220R CPU, 256GB RAM, and NVIDIA RTX 2080 Ti GPU. We use PyTorch (Paszke et al. 2019) as a deep learning framework.

D Visualization of Preemptively Robustified Examples

To understand how our preemptive robustification algorithm modifies each image, we tried increasing both the modification budgets (δ and ϵ) for visualization. Figure 6 shows the resulting examples. Interestingly, we find that our robustification process emphasizes human perceptible features of the original images, such as mountain peaks covered with snow or the fur color of the fox. Also, it tends to render colors much more vivid, and the object boundaries crisper. The results suggest that our preemptive robustification algorithm could also be applied to image synthesis tasks such as style transfer, super-resolution, or colorization, aligned with the findings of Santurkar et al. (2019).

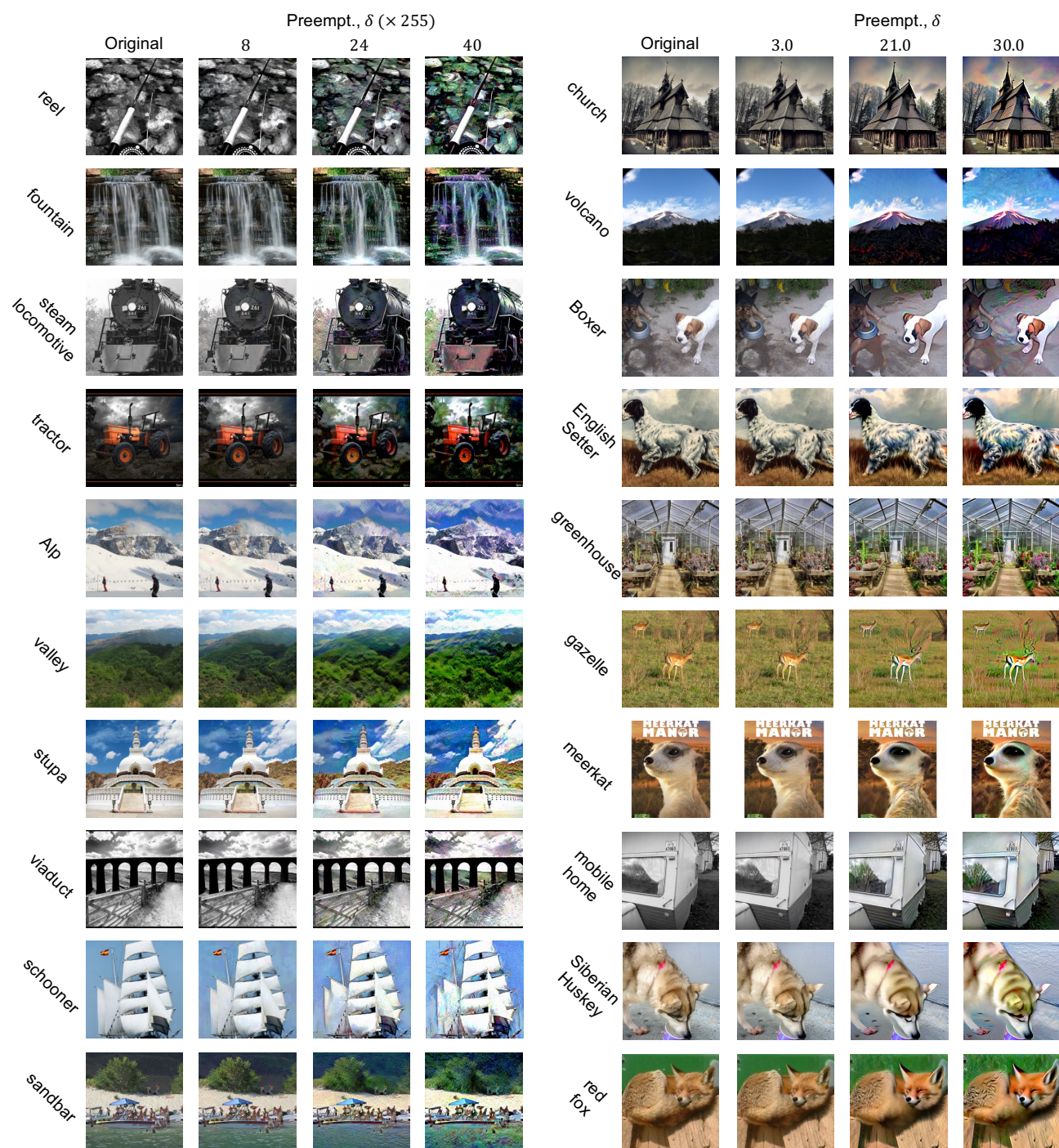


Figure 6: Examples of preemptively robustified images on ImageNet in ℓ_∞ (left) and ℓ_2 (right) settings.

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

Deokjae Lee¹ Seungyong Moon¹ Junhyeok Lee¹ Hyun Oh Song¹

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

¹Department 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 $s = [w_0, \dots, w_{l-1}] \in \mathcal{X}^l$ 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:

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

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:

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

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

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

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,

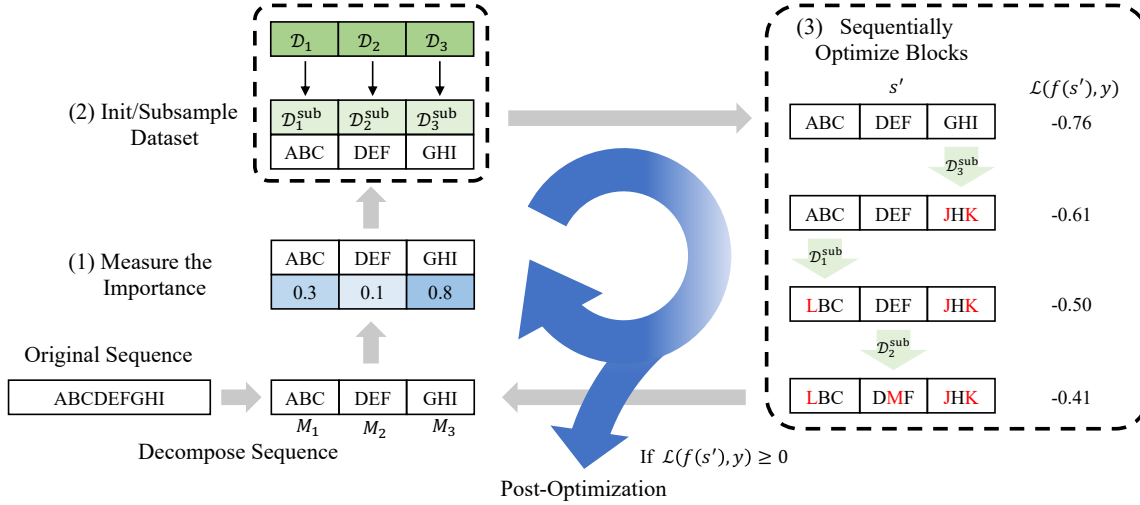


Figure 1: The overall process of BBA. A green arrow with a dataset $\mathcal{D}_k^{\text{sub}}$ denotes the Bayesian optimization step for the block M_k using $\mathcal{D}_k^{\text{sub}}$ as the initial dataset.

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 kernel¹ with automatic relevance determination (ARD) to automatically determine the degree to which each input dimension is important (MacKay, 1992). The kernel

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

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

Algorithm 1 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.
    $s_{\text{far}} \leftarrow \operatorname{argmax}_{s \in \mathcal{D} \setminus \mathcal{D}_{\text{sub}}} [\min_{s' \in \mathcal{D}_{\text{sub}}} d(s, s')]$ 
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 synonymy 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 se-

Algorithm 2 PostOpt(s , s_{adv} , $\mathcal{D}_{\text{sub}}$, N_{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  $s_{\text{adv}}$ .
    
```

quences 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.

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 $\text{EI}(x) = \mathbb{E}[\max(g(s) - g_{\mathcal{D}}^*, 0)]$, 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 $\text{Var}(g(B) \mid \mathcal{D})$ (Kulesza & Taskar, 2012). Concretely, we

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

(a) WordNet						(b) Embedding						(c) HowNet					
Dataset	Model	Method	ASR (%)	MR (%)	Qrs	Dataset	Model	Method	ASR (%)	MR (%)	Qrs	Dataset	Model	Method	ASR (%)	MR (%)	Qrs
AG	BERT-base	PWWS	57.1	18.3	367	AG	BERT-base	TF	84.7	24.9	346	AG	BERT-base	PSO	67.2	21.2	65860
		BBA	77.4	17.8	217			BBA	96.0	18.9	154			BBA	70.8	15.5	5176
	LSTM	PWWS	78.3	16.4	336		LSTM	TF	94.9	17.3	228		LSTM	PSO	71.0	19.7	44956
		BBA	83.2	15.4	190			BBA	98.5	16.6	142			BBA	71.9	13.7	3278
MR	XLNet-base	PWWS	83.9	14.4	143	MR	XLNet-base	TF	95.0	18.0	101	MR	XLNet-base	PSO	91.3	18.6	4504
		BBA	87.8	14.4	77			BBA	96.3	16.2	68			BBA	91.3	11.7	321
	BERT-base	PWWS	82.0	15.0	143		BERT-base	TF	89.2	20.0	115		BERT-base	PSO	90.9	17.3	6299
		BBA	88.3	14.6	94			BBA	95.7	16.9	67			BBA	90.9	12.4	403
	LSTM	PWWS	94.2	13.3	132		LSTM	TF	98.2	13.6	72		LSTM	PSO	94.4	15.3	2030
		BBA	94.2	13.0	67			BBA	98.2	13.1	54			BBA	94.4	11.2	138

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

(a) WordNet					(b) Embedding					(c) HowNet				
Dataset	Method	ASR (%)	MR (%)	Qrs	Dataset	Method	ASR (%)	MR (%)	Qrs	Dataset	Method	ASR (%)	MR (%)	Qrs
IMDB	PWWS	97.6	4.5	1672	IMDB	TF	99.1	8.6	712	IMDB	PSO	100.0	3.8	113343
	BBA	99.6	4.1	449		BBA	99.6	6.1	339		BBA	100.0	3.3	352
	LSH	96.3	5.3	557		LSH	98.5	5.0	770		LSH	98.7	3.2	640
	BBA	98.9	4.8	372		BBA	99.8	4.9	413		BBA	99.8	3.0	411
Yelp	PWWS	94.3	7.6	1036	Yelp	TF	93.5	11.1	461	Yelp	PSO	98.8	10.6	86611
	BBA	99.2	7.4	486		BBA	99.8	9.6	319		BBA	98.8	8.2	283
	LSH	92.6	9.5	389		LSH	94.7	8.9	550		LSH	93.9	8.0	533
	BBA	98.8	8.8	271		BBA	99.8	8.6	403		BBA	98.2	7.4	353

first select sequences with top- T acquisition values in the 1-Hamming distance ball $\mathcal{B}_H(s_D^*, 1)$ of the best sequence s_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 $\mathcal{B}_H(s, d_H(s, s_{adv}) - 1) \cap \mathcal{B}_H(s_{adv}, r)$, 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-mlab/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 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

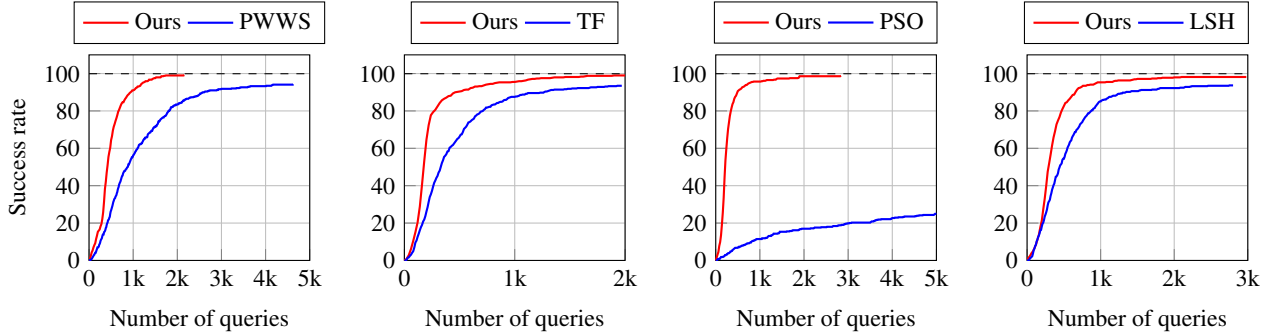


Figure 2: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the Yelp dataset. We use the HowNet based word substitution when comparing our method against LSH.

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 $E[d_H(s, s_{adv})/\text{len}(s)]$, 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 fewer 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.

Method	Level 0			Level 1			Level 2		
	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.

Model	Method	ASR (%)	MR (%)	Qrs	Both success	
					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.

Dataset	Method	BERT-base		LSTM	
		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

²For 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.

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.

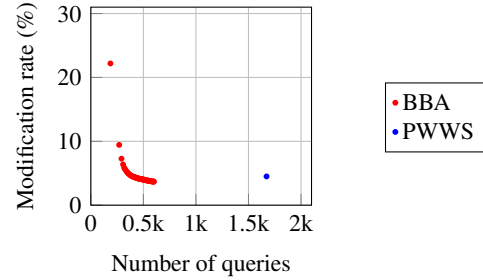


Figure 3: Modification rate versus the number of queries plot of adversarial texts generated by traversing N_{post} from 0 to 200 on the IMDB dataset against a BERT-base model. We use WordNet substitution method for the attack.

5.4.3. THE ACTUAL RUNTIME ANALYSIS

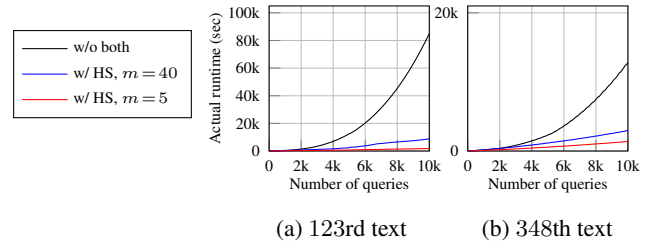


Figure 4: The cumulative runtime versus the number of queries plot. HS in the legend denotes history subsampling, and $m = k$ in the legend denotes block decomposition with the block size k .

To study the effectiveness of block decomposition and history subsampling techniques 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

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 <i>gorgeous</i> and exceptionally <i>unpolluted</i> ! 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 <i>awfully</i> clean! My only <i>grievances</i> is I <i>turned</i> there with my 2 small children and they were showing a very inappropriate R rated <i>footage</i> !	Negative
Protein Classification (EC50 level 0)		Label
Orig	MATPWRRALLMILASQVVTLVKCLEDDDVPEEWLLLVVQGQIGAGNYSYLRNLNHEGKIIIRMQSLRGDADLYVSDSTPHPSFDDYELQSVTCGQDVVSI PAHFQRPVIGIYGHPSHHESDFEMRVYYDRTVDQYPPGEAAAYFTDPTGASQQQAYAPEEAAQEEESVLWTILISILKLVLEILF	Non-Enzyme
BBA	MATPWRRALLMRLASQVVTLVKCLEDDDVPEEWLLLVVQGQIGAGNYSYLRNLNHEGKIIIRMQSLRGDADLYVSDSTPHPSFDDYELQSVTCGQDVVSI PAHFQRPVIGIYGHPSHHESDFEMRVYYDWTVDWYPPGEAAAYFTDPTGASQQQAYAPEEAAQEEESVLWTILISILKLVLEILF	Enzyme
TF	MATPWRRALLMILASQVVTLVKCLEDDDVPEEWLLLVVQGQIGAGNYSYLRNLNHEGKIIIRMQSLRGDADLYVSDSTPHPSFDDYELQSVTCGQDVVSI PAHFQRPVIGIYGHPSHHESDFEMRVYYDRTVDQYPPGEAWAYFCGWSGASQQQAYAPEEWWWFEEESVLDITILISGLKLVLEILF	Enzyme

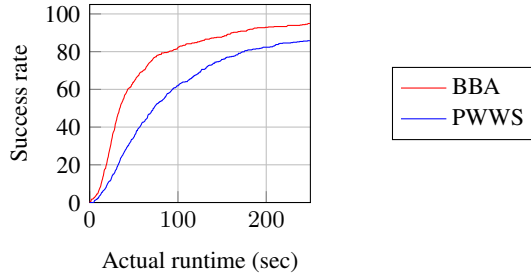


Figure 5: The cumulative distribution of the actual runtime required for the attack methods against the XLNet-large model on the Yelp dataset. Refer to Table 16 in Appendix D.1 for the detailed attack results.

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_θ 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 $k = 0, 1, \dots, l/m - 1$.
 - 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 $k = 0, 1, \dots, l/m - 1$.
 - 5: **for** ITER = 0 **to** $R - 1$ **do**
 - 6: Sort the block indices $[0, \dots, l/m - 1]$ to $[\gamma_0, \dots, \gamma_{l/m-1}]$ in order of decreasing importance score.
 - 7: **for** k **in** $[\gamma_0, \dots, \gamma_{l/m-1}]$ **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_{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: Greedy 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_{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

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

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

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

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

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

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).

Dataset	$\mathcal{C}$	WordNet	Embedding	HowNet
	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).

Dataset	$\mathcal{C}$	WordNet		Embedding		HowNet	
	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).

Dataset	$\mathcal{C}$	WordNet	Embedding
	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).

Dataset	$\mathcal{C}$	WordNet	Embedding	HowNet
	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 classification 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 $\mathcal{O}(N)$ 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 $\mathcal{O}(N^3 + N^2 C m)$ 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 $\mathcal{O}(N|\mathcal{D}_k^{\text{sub}}|)$ 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 $\mathcal{O}(Nn + N^3 l^3 / m^3)$ 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 sen-

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.

tences in QNLI.

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 Strodtz 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
IMDB	LSTM	242.5	78.6
	XLNet-large		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 sentence-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						(b) Embedding					
Dataset	Model	Method	ASR (%)	MR (%)	Qrs	Dataset	Model	Method	ASR (%)	MR (%)	Qrs
IMDB	XLNet-large	PWWS	98.8	5.7	1697	IMDB	XLNet-large	TF	99.0	8.8	789
		BBA	99.8	5.4	573			BBA	99.6	6.9	638
	BERT-large	PWWS	98.1	4.9	1661		BERT-large	TF	97.2	7.0	660
		BBA	100.0	4.6	619			BBA	98.7	6.4	559
Yelp	XLNet-large	PWWS	94.5	10.8	1107	Yelp	XLNet-large	TF	95.9	16.7	635
		BBA	98.2	9.4	485			BBA	99.4	12.6	475
	BERT-large	PWWS	75.1	7.7	1206		BERT-large	TF	68.8	14.1	924
		BBA	94.7	7.7	657			BBA	93.9	11.3	548
MR	XLNet-large	PWWS	82.7	14.9	145	MR	XLNet-large	TF	94.6	18.4	103
		BBA	87.6	14.8	98			BBA	96.9	16.2	68
	BERT-large	PWWS	80.0	14.6	146		BERT-large	TF	90.8	18.7	111
		BBA	85.1	14.2	110			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 .

m	ASR (%)	w/o Post-optimization		w/ Post-optimization	
		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 fewer 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.

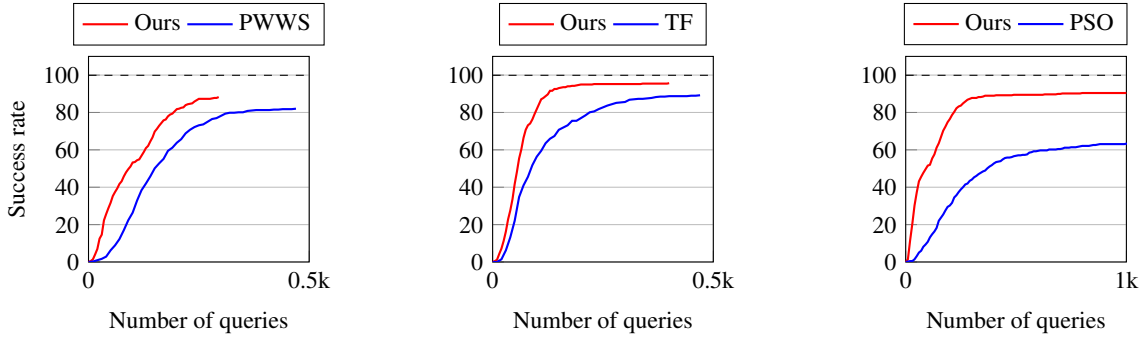


Figure 6: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the Movie Review dataset.

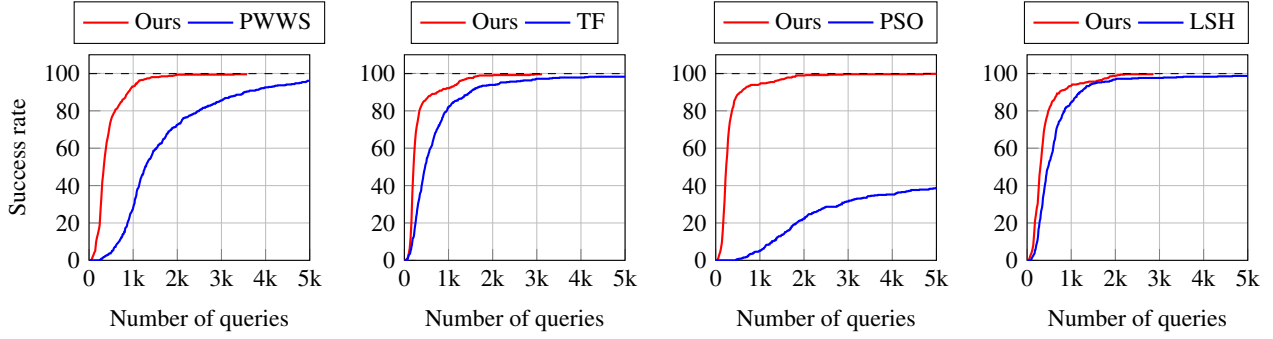


Figure 7: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the IMDB dataset. We use the HowNet based word substitution when comparing our method against LSH.

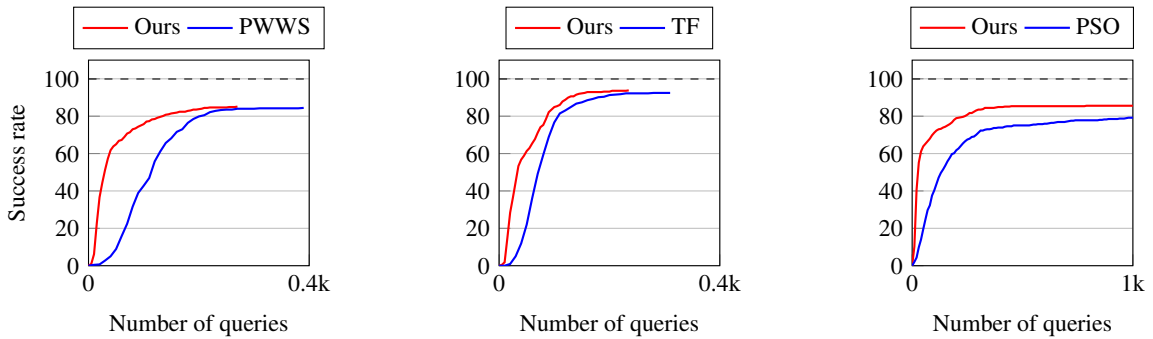


Figure 8: The cumulative distribution of the number of queries required for the attack methods against a BERT-base model on the MNLI dataset.

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	<i>undergo</i> from a decided <i>dearth</i> 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 <i>allright massaging</i> that I've ever had.	Negative
TF	I had never been here before, but I'm <i>excited me</i> tried it out. <i>His</i> was the <i>alright</i> massage that I've ever <i>received</i> .	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 <i>ask</i> were returned when I left a few VM's. Will <i>refreshing</i> this if they ever do call back.	Positive
TF	Not sure if they are closed for <i>companies</i> but none of my <i>appeals</i> were returned when I <i>going a short</i> VM's. <i>Dedication</i> 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." <i>Unluckily</i> , 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 <i>honest</i> 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 <i>documentary</i> , because I noticed the cover in the video rental store. I <i>snapped</i> Nolte, Connely, Madsen, 40's time setting, and thought "hmm, can't be too bad." <i>Unluckily</i> , after watching it, my <i>career</i> was "not too good". Its kind of a Chinatown ripoff, but the <i>seediest farewell</i> is that other than Nolte, the other members of the <i>world</i> didn't get <i>substantial</i> screen time. But its a <i>honest ballad</i> to see once I guess. And Melanie's role was <i>humble</i> enough that she wasn't given a <i>opportunity</i> to be a <i>headache</i> .	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 <i>dispute</i> what color one wears.	Entailment
TF	<i>His</i> doesn't <i>do</i> a difference what <i>colourful</i> one <i>focuses</i> .	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, <i>invoking</i> as he was admitted to the <i>hostels</i> .	Neutral
TF	I turned a curve and was just in time to see him <i>cyclic</i> the big brass <i>chime</i> , <i>noting</i> as he was <i>hospitalised</i> to the <i>sarcophagus</i> .	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 <i>journals</i> , 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 <i>writs</i> to <i>activation</i> enough <i>timing</i> 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