How AE Searching Acceleration Benefits PGD Attack and Adversarial Training on Text Classifiers

Xuanzhou Chen
Georgia Institute of Technology
xchen920@gatech.edu

Zhangqi Luo
Georgia Institute of Technology
jamesluo@gatech.edu

Abstract

Adversarial training is generally very slow, which requires adversarial example generation per epoch on the fly. In order to fasten the adversarial example searching process, we proposed to an acceleration method utilizing k-means and modified locality sensitive hashing. We applied it on both sides - attacker (Projected Gradient Attack) and defender (adversarial training). We proposed a novel adversarial training method by leveraging the acceleration. In experiments, we evaluate the performance of accelerated PGD attack with the original attack [SDF22] as well as our newly designed adversarial training with previous work on fast adversarial training, such as FreeAT[Sha+19], YOPO[Zha+19] and FreeLB[Zhu+20]. In addition, our adversarial training should avoid robustness overfitting and catastrophic overfitting[KLL21] by algorithm design and we will further investigate it in experiments. Github Link: https://github.com/xchen793/PGD-attack.

1 Related Work

PGD Attack in NLP. Different from computer vision, Projected Gradient Attck(PGD attack) in Natural Language Processing includes two steps: Gradient Step and Projection Step. After finding the optimizer in expanded continuous space in Gradient Step. The optimizer would be projected into discrete space to closest embedding vectors by comparing the cosine similarity. Overall, PGD attack we mentioned in this project a is black-box embedding-level substitution textual attack.

Adversarial Training. In computer vision, adversarial training(AT) was originally proposed as a means to enhance the security of machine learning systems and tested on MNIST dataset [GSS15]. However, K-PGD adversarial training [Mad+18] is generally slow since it requires additionally k forward and k backward passes through the network to compute the adversarial version of the training batch, compared to natural training. Many researchers were dedicated to cut down the adversarial training time. [Sha+19] proposed free adversarial training(freeAT), a 3-30 times faster adversarial training algorithm that facilitates model robustness compared to K-PGD adversarial training method proposed by [Mad+18]. In natural language processing, relevant work includes adversarial training through data augmentation, regularization or GANs, virtual adversarial training and robustness through human intervention [Goy+23]. Since previous work has not been applied to PGD attack in NLP domain, a novel fast and effective adversarial training algorithm is needed. Specifically, we focused on exploring adversarial training by data augmentation in this project.

Catastrophic Overfitting. One drawback of freeAT is that the minibach replay may cause catastrophic forgetting and generalization error increases as m grows. Catastrophic forgetting is a challenge when focusing to make adversarial training computationally efficient, the robust accuracy with respect to a PGD adversarial suddenly and drastically drops to 0% on the training data. It is necessary to design a fast adversarial training to avoid such catastrophic overfitting.

Robustness Overfitting. Different from the standard setting of deep networks, overfitting for adversarially robust training can result in worse test set performance proposed by [Don+21] implicitly imposes overfitting problem by iteratively adding adversarial examples in training dataset during each epoch. Investigation and design caution are needed to avoid robustness overfitting in our PGD adversarial training method, which refers to generate adversarial examples using data augmentation.

Equipped with this perspective, we make the following contributions.

1. We adopted k-means clustering to pre-

process token embeddings for Projected Gradient Descent(PGD) Attack. Our attack is 10 times faster than the block-sparse textual attack proposed by [SDF22], with the total time complexity in "method" section. 2. We designed a novel adversarial training for corresponding PGD attack, and compare its effectiveness with previous fast Adversarial training methods including FreeAT[Sha+19], YOPO[Zha+19], FreeLB[Zhu+20]. 3. We further investigated whether robustness overfitting/catastrophic overfitting exists in our adversarail training method.

2 Methods

Data Pre-processing Acceleration. Our first step is to accelerate the adversarial example process using iterative k-means pre-processing(IKMP), which iteratively groups the high-dimensional embedding vectors in the vocabulary set into cluster centroids according to cosine similarity. The pre-processing time complexity is computed as $T = O((K+1) \times |\mathcal{V}| \times \texttt{maxIter} + \sum_i^{\texttt{maxIter}} N_i \times \texttt{dim}) + (N + \frac{|\mathcal{V}|}{N})$, where N_i is the number of "buckets" in i-th iteration and N is the final number of "buckets.

Accelerated AE Generation with PGD attack. After the vocabulary set is partitioned into different clusters with each represented by an average token embedding vectors, we calculate the cosine similarity to find the average token embedding vectors most close to the clean token x_i. Then search the corresponding bucket to find most similar token as our perturbed token x_i'. [See Figure 1]

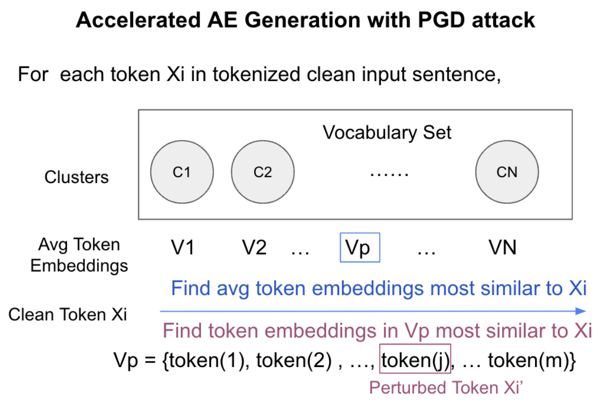
Accelerated AE Generation with PGD attack
For each token Xi in tokenized clean input sentence,
Vocabulary Set
Clusters
C1
C2
......
CN
Avg Token Embeddings
V1
V2
...
Vp
...
VN
Find avg token embeddings most similar to Xi
Clean Token Xi
Find token embeddings in Vp most similar to Xi
Vp = {token(1), token(2) , ..., token(j), ... token(m)}
Perturbed Token Xi'

Figure 1: AE Generation in PGD Attack

Accelerated AE Generation in Adversarial Training. During adversarial training, we adopted IKMP to cluster the clean training dataset and assigned each cluster $C_i, i \in \{1, ..., N\}$ with average token embedding vectors as a representation $V_i, i \in \{1, ..., N\}$, then retrieved M samples from each cluster C_i to from a corresponding M-size sub-cluster d_i. For each input sentence embedding in a sub-cluster, we generate adversarial examples using accelerated AE generation with PGD attack. Once we have N adversarial example(AE) clusters, we can construct our adversarial training dataset D_{adv} by adding them together. At the end, D_{adv} would be used in adversarial training. [See Figure 2]

Advantages by Design. We further claimed that there are several advantages by this design:

1. Parallel structure due to identical and independent process in each thread allows parallel computing;
2. The original search space is truncated into subspaces to perform PGD attack allows faster search;
3. Avoid catastrophic overfitting from source, adversarial examples generated by PGD attack will not suffer from this drawback of one-single adversarial attack;
4. Compared to A2T[YQ21], smaller sub datasets are used to generate adversarial examples in each thread, which will enable adversarial training on LLMs;
5. Adversarial examples generated are more evenly distributed in semantic space, i.e., clustering and sampling helps reduce the model inherent bias.

3 Experiments

We conducted experiments to better understand the multiple facets of adopting IKMP to accelerate in adversarial example searching and the overall impact on both PGD attack and our adversarial training method.

Comparisons. There are two comparison sets: 1) Overall performance of our PGD attack(with IKMP) vs. original PGD attack[SDF22] and other fast textual attacks. 2) Overall performance of our adversarial training vs. other fast adversarial training methods, including FreeAT[Sha+19], YOPO[Zha+19], FreeLB[Zhu+20]. Both experiments were carried out on text classification tasks.

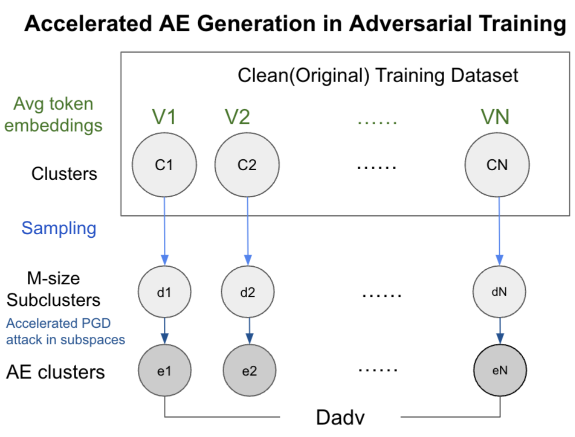
Accelerated AE Generation in Adversarial Training
Clean(Original) Training Dataset
Avg token embeddings
V1
V2
VN
Clusters
C1
C2
CN
Sampling
M-size Subclusters
d1
d2
dN
Accelerated PGD attack in subspaces
AE clusters
e1
e2
eN
Dadv

Figure 2: AE Generation in Adversarial Training

Models & Tasks. We selected GPT-2[Rad+19], BERT[Dev+19] and RoBerta[Liu+19] as our LLM backbones. All three language models were fine-tnued and aligned on the yelp-polarity-reviews, Ag-News[ZZL15] and MultiNLI[WNB18] datasets for classification. In PGD attack acceleration, we only adopted GPT-2 as the victim model.

Datasets. The Yelp reviews polarity dataset for binary sentiment classification is constructed by considering stars 1 and 2 negative, and 3 and 4 positive. For each polarity 280,000 training samples and 19,000 testing samples are taken randomly. In total there are 560,000 trainig samples and 38,000 testing samples. Negative polarity is class 1, and positive class 2. AG is a collection of more than 1 million news articles gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. The AG's news topic classification dataset is constructed by choosing 4 largest classes from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and testing 7,600. This dataset is designed for research purposes such as data mining (clustering, classification, etc) and information retrieval (ranking, search, etc). The Multi-Genre Natural Language Inference (MultiNLI) dataset has 433K sentence pairs. MultiNLI offers ten distinct genres (Face-to-face, Telephone, 9/11, Travel, Letters, Oxford University Press, Slate, Verbatim, Goverment and Fiction) of written and spoken English data with 3-class: entailment, neural and contradiction.

Metrics. We also used multiple metrics: successful attack rate(SAR) are used to measure the attack effectiveness, which computed as $\frac{\#\text{successful attack}}{\#\text{total attack}}$; token error rate(TER) are used to measure perturbing magnitude by calculating the similarity difference between perturbed text and clean text; average time over 200 attacks are used to show the attack speed.

Algorithm 1 Iterative K-means Preprocessing (IKMP)

Input: maxIter - max iteration number; $|\mathcal{V}|$ - total number of vector embeddings $\vec{e}_i \in \mathcal{E}_\mathcal{V}$, $i = 1, 2, ..., |V|$; K - initial number of clustroids
Output: centroids $\leftarrow [\vec{v}_1, ..., \vec{v}_K]$; buckets $\leftarrow$ dict{'idx': [],...}

```
1: $iter \leftarrow 0$
2: centroids $\leftarrow [\vec{v}_1, ..., \vec{v}_K]$
3: while $iter <$ maxIter do
4:     buckets $\leftarrow$ dict{}
5:     for i in $1, ..., |\mathcal{V}|$ do
6:         for j in $1, ..., K$ do
7:             idx $\leftarrow argmin_j \frac{\vec{e}_i^T \cdot \vec{v}_j}{||\vec{e}_i||_2 \cdot ||\vec{v}_j||_2}$
8:             if idx not in bucket.keys then
9:                 buckets[idx] = list[]
10:            end if
11:            buckets[idx].append($\vec{e}_i$)
12:        end for
13:    end for
14:    centroids $\leftarrow$ list[]
15:    for key in buckets.keys do
16:        $v \leftarrow$ AvgVector(buckets[key])
17:        centroids.append($v$)
18:    end for
19:    $iter = iter + 1$
20:    if all AvgVectors converge then
21:        break
22:    end if
23: end while
```

Note*: In the implementation of verifying the average vector convergence, we use the summation of $l2$ norm to calculate the difference between each average vectors in the previous iteration and the current iteration.

Harware Environment. The hardware we use for training and testing model is NVidia RTX A5000 GPU with 24 GB CUDA memory. The GPU is manually allocated before every experiment. The GPUs we use are those with no other existing activites. It allows accurate measurement and more stable training process. Total GPU memory is $\sim$ 24G.

Hyperparameters. One hyperparameter α in total loss [SDF22] is scheduled during the AE acceleration with values in [10, 8, 5, 2].

Experimental Settings. During Gradient Descent in PGD attack, we adopted adam as our optimizer, learning rate at 0.05, max iteration number is 300. The initial cluster centroids $K = 32$ in IKMP. K-means max iteration number is 100.

3.1 Preliminary Results

In the context of text classification on Yelp-polarity reviews[ZZL15], we conducted a comparison on both attack time and successful attack rate of our IKMP-accelerated PGD attack and the original PGD attack[SDF22], as well as the A2T-attack[YQ21]. This evaluation was carried out utilizing GPT-2[Rad+19] and BERT[Dev+19] and the preliminary results are presented in Table 1.

Attack time comparison Based on the experiment result in Figure 3, we found that bucket-based adversarial attack is at least 10 times faster than the original block-space adversarial attack when tested on GPT-2-based text classifier. In Table 1, T(avg) GPT-2 denotes time(s) per attack when classification model is trained based on GPT-2. T(avg) BERT denotes time(s) per attack when such model is trained based on BERT. We can see that when attacking GPT-2-based text classifier. adding clustering steps before PGD also allows performance win over the attack in A2T in term of attacking speed. Note that on BERT-based classifier, PGD attack with clustering is slower. We suspect that A2T attack has lower optimization in its implementation. For next step, we will refactor our attacking algorithm with the framework used in A2T attack and see if we can take advantage of existing optimizations.

Attack	**T(avg) GPT-2**	**T(avg) BERT**
A2T attack	2.7s	2.9s
$\mathrm{PGD}_{\mathrm{IKMP18}}$	1.29s	5.40s

Table 1: Attack Time Comparison

SAR comparison SAR demote successful attack rate. It shows the percentage of attack attempts that successfully fools the classifier as expected. We measure the SAR of PGD attack with and without clustering mechanism on GPT-2 based text classifier. We found that SAR for PGD without clustering is slightly better. (Table 2) That is not surprising because original PGD has a larger space for searching the embeddings. It is a tradeoff for accelerated PGD. Further, we case study the SAR of attack method introduced in A2T. We test it on a BERT-based classifier finetuned with Yelp polarity datasets. We found that its SAR is only 0.37. It shows that the one-step AE generation is inherently unstable.

Attack	**SAR**
$\mathrm{PGDattack}_{\mathrm{IKMP18}}$	0.93
PGDattack	0.995

Table 2: SAR Comparison

Memory Profiling. Adversarial attack is known for utilizing a nontrivial amount of GPU memory. To investigate the GPU memory usage of bucket-based attack, we measure the CUDA memory allocation of the original and new algorithm through pytorch profiler. The original attack requires 344 MB CUDA memory allocation for computational kernels and other tensors, while our attack needs 340 MB CUDA memory. We suspect that observed memory saving is due to smaller searching space of adversarial embedding. However, before bucket-based attack, 8 MB CUDA memory is used for doing k-mean and forming buckets. It means overall CUDA memory allocation for new approach is 348 MB. It's slightly more than the old approach, but such trade-off is acceptable given how much faster the new way is.

Adversarial input quality evaluation. In A2T attack, there is no regularization in similarity between original input and adversarial input, while in the loss function of PGD attack, we leverage both label likelihood and text similarity. Excluding text similarity from attacking helps speed up the AE generation in adversarial training introduced in A2T. However, the adversarial input might look dramatically different original one. To verify our concerns, we encode the original sentences and adversarial sentences through google's universal sentence encoder. We then check the cosine similarity between two sentences. We found that the generated adversarial examples are actually in decent quality. It inspires us to try a improved hyperparameter for PGD attack with clustering in the future and relax the constraint in similarity between original input and adversarial input.

4 Contribution

We as a group of two make collaborative contribution to this project:

1. PGD attack acceleration on GPT2 as text clas-

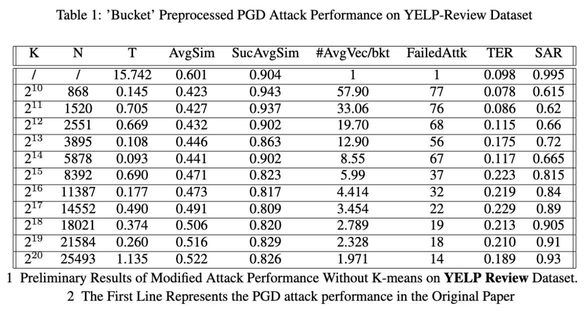

Table 1: 'Bucket' Preprocessed PGD Attack Performance on YELP-Review Dataset

K	N	T	AvgSim	SucAvgSim	#AvgVec/bkt	FailedAttk	TER	SAR
/	/	15.742	0.601	0.904	1	1	0.098	0.995
2^{10}	868	0.145	0.423	0.943	57.90	77	0.078	0.615
2^{11}	1520	0.705	0.427	0.937	33.06	76	0.086	0.62
2^{12}	2551	0.669	0.432	0.902	19.70	68	0.115	0.66
2^{13}	3895	0.108	0.446	0.863	12.90	56	0.175	0.72
2^{14}	5878	0.093	0.441	0.902	8.55	67	0.117	0.665
2^{15}	8392	0.690	0.471	0.823	5.99	37	0.223	0.815
2^{16}	11387	0.177	0.473	0.817	4.414	32	0.219	0.84
2^{17}	14552	0.490	0.491	0.809	3.454	22	0.229	0.89
2^{18}	18021	0.374	0.506	0.820	2.789	19	0.213	0.905
2^{19}	21584	0.260	0.516	0.829	2.328	18	0.210	0.91
2^{20}	25493	1.135	0.522	0.826	1.971	14	0.189	0.93

1 Preliminary Results of Modified Attack Performance Without K-means on **YELP Review** Dataset.
2 The First Line Represents the PGD attack performance in the Original Paper

Figure 3: Accelerated PGDattack Performance on Yelp with Different K

sifier, including algorithm design, implementation and experiments. (Xuanzhou Chen) 2. Related Work (Xuanzhou Chen) 3. Method (Xuanzhou Chen) 3. Code Implementation (Xuanzhou & Zhangqi) 4. Run experiments on previous adversarial training as baseline. (Zhangqi Luo) 4. Profiling on memory usage of original attacks(Zhangqi Luo) 5. Experiment & Results (Xuanzhou & Zhangqi)

References

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML].

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 649–657.

[Mad+18] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

[WNB18] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. URL: http://aclweb.org/anthology/N18-1101.

[Dev+19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[Liu+19] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[Rad+19] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[Sha+19] Ali Shafahi et al. *Adversarial Training for Free!* 2019. arXiv: 1904.12843 [cs.LG].

[Zha+19] Dinghuai Zhang et al. *You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle*. 2019. arXiv: 1905.00877 [stat.ML].

[Zhu+20] Chen Zhu et al. *FreeLB: Enhanced Adversarial Training for Natural Language Understanding*. 2020. arXiv: 1909.11764 [cs.CL].

[Don+21] Xinshuai Dong et al. *Towards Robustness Against Natural Language Word Substitutions*. 2021. arXiv: 2107.13541 [cs.CL].

[KLL21] Hoki Kim, Woojin Lee, and Jaewook Lee. "Understanding catastrophic overfitting in single-step adversarial training". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 9. 2021, pp. 8119–8127.

[YQ21] Jin Yong Yoo and Yanjun Qi. *Towards Improving Adversarial Training of NLP Models*. 2021. arXiv: 2109.00544 [cs.CL].

[SDF22] Sahar Sadrizadeh, Ljiljana Dolamic, and Pascal Frossard. *Block-Sparse Adversarial Attack to Fool Transformer-Based Text Classifiers*. 2022. arXiv: 2203.05948 [cs.CL].

[Goy+23] Shreya Goyal et al. *A Survey of Adversarial Defences and Robustness* *in NLP*. 2023. arXiv: 2203.06414 [cs.CL].