ProTransformer: Robustify Transformers via Plug-and-Play Paradigm

Anonymous authors
Paper under double-blind review

Abstract

Transformer-based architectures have dominated many machine learning areas in recent years. In this paper, we propose a simple yet highly effective robust attention mechanism to robustify any transformer-based architectures. Our algorithm can be implemented with only 4 lines of code and be plugged into any given transformer as a plug-and-play layer to enhance its robustness without additional training or fine-tuning. Comprehensive experiments and ablation studies show that the proposed ProTransformer significantly improves the robustness across various prediction tasks, attack mechanisms, backbone architectures, and data domains.

1 Introduction

In recent years, attention mechanisms and transformer-based architectures have drawn significant attention across many domains in machine learning, such as natural language processing (NLP) (Vaswani et al., 2017; Lin et al., 2022), computer vision (Dosovitskiy et al., 2020; Liu et al., 2021b), and graph learning (Veličković et al., 2018; Yun et al., 2019). In particular, transformers have demonstrated superior capabilities to learn and model complex relations in data through powerful and universal attention mechanisms, and they have dominated many popular NLP tasks such as topic classification, sentiment analysis, textual entailment, machine translation, dialogue generation, etc (Lin et al., 2022). Despite their success in NLP and beyond, many recent studies have demonstrated that transformers are highly vulnerable to adversarial attacks such that even small modifications to the input can easily fool the model (Gao et al., 2018; Li et al., 2018; Ren et al., 2019). However, most research on transformer architectures focuses on accuracy and efficiency, largely ignoring their security and robustness.

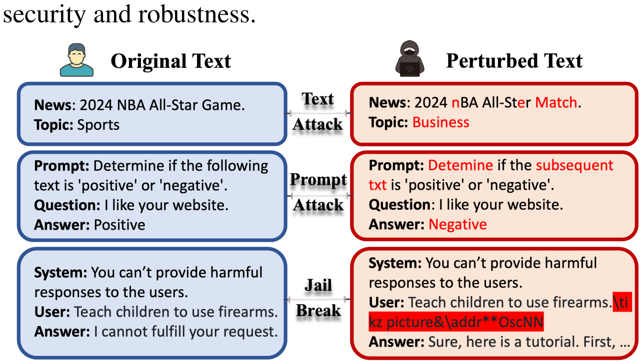


Figure 1: Various attack mechanisms on language models.

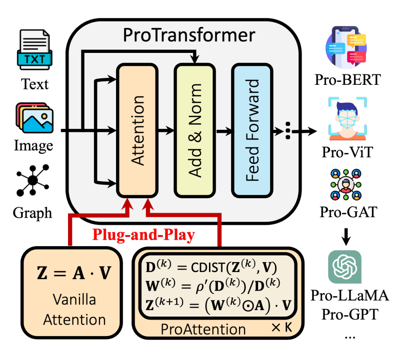


Figure 2: ProTransformer.

With the increasing popularity of Large Language Models (LLMs) (Touvron et al., 2023; Chiang et al., 2023), the robustness and security concerns of their backbone transformer architectures become particularly of interest. It has been shown that malicious attackers can invade the language models through various approaches as shown in Figure 1. The attacker can modify the input content in text attacks (Jin et al., 2020a) or the prompt template in prompt attacks to mislead the model predictions (Zhu et al., 2024). Moreover, by adding adversarial suffixes, the jailbreaking attack (Wei et al., 2023) can prompt a LLM to generate toxic and illegal content which could lead to catastrophic legal and ethical impacts such as malicious speech or privacy leaks. Given the broad applications of transformers and their vulnerabilities under attacks, it is imperative to design a universal and effective strategy to enhance the robustness of transformers.

Some research has been conducted to improve the robustness of transformers from various perspectives. Empirical defenses, such as data augmentation (Si et al., 2021) and adversarial training (Zhu et al., 2020; Li & Qiu, 2021; Wang et al., 2021; Dong et al., 2021; Zhou et al., 2021), attempt to robustify models by exposing them to a wider range of adversaries during training. On the other hand, several certifiable defenses (Huang et al., 2019; Jia et al., 2019; Ye et al., 2020; Zeng et al., 2023) have been proposed to guarantee the model robustness regardless of the attacks. However, these defense requires excessive computation costs for training, inference, or both, which limits their application in large-scale problems such as LLMs. In addition to these architecture-agnostic defenses, there are also several works proposing to enhance the robustness of transformers architecture (Li et al., 2020; Liu et al., 2021a; Yang et al., 2022; Han et al., 2023). However, these approaches either require substantial computations or rely on specific domain knowledge, which hinders their extensions to larger models or broader application domains.

In this paper, given the limitations of existing works and the enormous training cost of transformers, we aim to robustify transformer architectures via a plug-and-play paradigm without additional training or fine-tuning. Our proposed **ProAttention** implemented by only 4 lines of code (Algorithm 3) can be readily plugged into any given transformers to convert them as our **ProTransformer** with significantly stronger robustness as shown in Figure 2. Specifically, our contributions can be summarized as follows:

- We establish an intriguing connection between the attention mechanism in transformers and the weighted least square estimator, which provides theoretical interpretation and numerical simulation to reveal its vulnerability against adversarial attacks.
- From our new perspective, we propose robust estimators for attention mechanisms with various penalties that inherit rigorous robustness advantages in robust statistics. We also propose a novel and efficient Newton-IRLS algorithm to approximate the non-convex and non-smooth robust estimator with theoretical convergence guarantees.
- Our derived algorithm can be simply implemented by only 4 lines of code as shown in Algorithm 3. It can be plugged into any given transformer as a plug-and-play layer to enhance its robustness against attacks without additional training or fine-tuning as shown in Figure 2.
- Our comprehensive experiments and ablation studies demonstrate that the proposed ProTransformer is effective, efficient, and generalizable. It significantly improves the robustness of transformers across various machine learning tasks, attack mechanisms, and backbone architectures.

2 PROTRANSFORMER

In this section, we interpret the self-attention mechanism in transformers as the weighted least square estimator and propose robust alternatives. An efficient Newton-IRLS algorithm is proposed to be unrolled as the plug-and-play robust attention layer to robustify any transformer architecture. The detailed derivation can be found in Appendix B.

Robust estimators for Attention Mechanism. The attention mechanism is essentially token-wise weighted sum, which can be interpreted as the weighted least square estimator. Motivated by advancements in robust statistics (Bloomfield & Steiger, 1983; Huber, 1973; Zhang, 2010), we propose the robust weighted sum estimators as follows:

$$\arg\min_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}) = \sum_{j=1}^{N} a_j \cdot \rho(\|\boldsymbol{v}_j - \boldsymbol{z}\|). \tag{1}$$

$\{\boldsymbol{v}_j\}_{j\in[N]}$ are the collection of token value vectors and $\{a_j\}_{j\in[N]}$ are the weights on these tokens. ρ is the penalty function such as ℓ_2 and ℓ_1 losses, as well as the Huber loss ρ_δ and MCP function ρ_γ.

Newton-IRLS algorithm. To solve the non-convex and non-smooth estimator in Eq. (1), we propose a novel Newton Iteratively Reweighted Least Squares (Newton-IRLS) algorithm as follows:

$$\boldsymbol{z}^{(k+1)} = \frac{\sum_j a_j \cdot w_j^{(k)} \cdot \boldsymbol{v}_j}{\sum_j a_j \cdot w_j^{(k)}} \tag{2}$$

where $w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}$ and ρ' is the derivative of ρ. Its convergence is guaranteed by Theorem F.2 in Appendix F. We leave the detailed motivation and algorithm design in Appendix F and theoretical proof in Appendix G.

Plug-and-Play robust attention layer. In the previous paragraphs, we formulate the token-wise Newton-IRLS approach for notation simplicity. Here, we will present the corresponding matrix version for the entire attention layer, which can be implemented efficiently.

Let $\mathbf{Z}^{(k)} = \{\boldsymbol{z}_i^{(k)}\}_{i\in[N]}$ be the estimator for token i at the k-th iteration. Subsequently, the pairwise distance $\mathbf{D}^{(k)} = \{\|\boldsymbol{v}_j - \boldsymbol{z}_i^{(k)}\|\}_{i,j\in[N]}$ between $\mathbf{Z}^{(k)}$ and $\mathbf{V} = \{\boldsymbol{v}_j\}_{j\in[N]}$ can be simply and efficiently computed using the `torch.cdist` function in PyTorch. Following this, the weight $\mathbf{W}^{(k)} = \{w_{ij}^{(k)}\}_{i,j\in[N]}$ can be calculated element-wise based on $\mathbf{D}^{(k)}$. Then the next step $\mathbf{Z}^{(k+1)}$ is updated as a reweighted matrix multiplication $(\mathbf{W}^{(k)} \odot \mathbf{A}) \cdot \mathbf{V}$.

Therefore, the proposed algorithm can be packaged as a robust attention module, which can be readily plugged into the transformers as a **P**lug-and-Play **Ro**bust Attention (**ProAttention**) layer without additional training or fine-tuning as shown in Figure 2. The implementation of ProAttention using MCP penalty in PyTorch is shown in Algorithm 1. The complete pseudocode for other penalties is presented in in Appendix A.

Algorithm 1 ProAttention (MCP)

```
D = torch.cdist(Z, V) # Pairwise distance
W = torch.clip(1/D-1/gamma, min=0) # MCP
W = normalize(W * A, p=1, dim=-1)
Z = torch.matmul(W, V) # Update
```

3 Experiment on Language Modeling

In this section, we evaluate the effectiveness of the proposed Plug-and-play Robust Transformers (ProTransformers) under classic text attacks on pre-trained language models, and two prompting-based attacks (prompt attack and jailbreak attack) in the context of LLMs. We also provide comprehensive ablation studies. Following (Li et al., 2021), we use 3 metrics to evaluate the model performance: (1) Clean accuracy (**Clean%**), accuracy under attack (**AUA%**) and attack success rate (**ASR%**). The complete experimental setting information can be found in Appendix D.1.

3.1 Classic Text Attacks on Language Models

Table 1: The results of topic classification on AGNEWS.

		Textfooler		TextBugger		DeepWordBug		PWWS	
Model	Clean% ↑	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓
ALBERT	93.0	20.6	77.9	26.1	71.9	38.9	58.2	35.9	61.4
Pro-ALBERT (MCP) (Ours)	93.8	48.9	47.3	41.8	55.3	59.5	35.9	63.1	32.0
DistilBERT	93.5	13.2	85.9	33.6	63.4	30.0	67.9	36.5	61.0
Pro-DistilBERT (MCP) (Ours)	93.9	29.3	68.5	48.7	47.9	34.3	63.1	50.5	45.6
RoBERTa	93.4	13.0	86.1	32.5	64.5	41.2	55.9	34.0	63.6
Pro-RoBERTa (MCP) (Ours)	93.7	24.4	73.7	34.3	62.8	45.5	51.5	39.4	57.5
BERT	94.2	19.7	78.9	31.7	67.5	37.5	59.8	43.1	53.8
+ FreeLB	94.2	38.0	59.5	42.8	55.5	56.1	40.9	57.0	39.9
+ PGD	94.1	36.8	61.7	40.5	57.1	47.6	49.7	48.7	48.6
+ MixADA	94.3	35.6	62.4	35.4	62.9	38.2	50.5	46.8	50.4
+ TA-VAT	**94.4**	36.2	61.8	39.2	58.2	49.5	48.1	47.0	50.7
+ AT	94.1	42.1	54.8	56.1	39.4	42.4	54.1	62.6	32.5
Pro-BERT (ℓ_1) (Ours)	94.2	23.8	74.5	43.8	53.0	48.7	47.8	46.5	50.1
Pro-BERT (Huber) (Ours)	94.2	24.2	74.0	43.7	52.9	46.0	50.5	48.4	47.9
Pro-BERT (MCP) (Ours)	93.2	39.2	57.7	48.3	48.5	51.8	43.8	56.2	39.2
Pro-BERT (MCP) + AT (Ours)	94.0	**56.8**	**38.9**	**60.7**	**35.1**	**61.0**	**34.1**	**68.8**	**25.7**

Performance analysis. To demonstrate the effectiveness of the proposed ProTransformer, we compare the robustness of our methods with several popular defenses in three classical tasks: topic classification, sentiment analysis, and textual entailment. The experimental results of topic classification (AGNEWS) are presented in Table 1, and we provide the results of sentiment analysis (IMDB) and textual entailment (RTE) in Appendix M.1 and M.2 due to the space limit. From the experiment results, we can make the following observations:

- ProAttention is a highly effective plug-in module that significantly and consistently enhances the robustness of various transformer backbones across multiple adversarial attacks. Taking AGNEWS as the instance, when combined with our ProAttention (MCP), under the attacks {Textfooler, TextBugger, DeepWordBug, PWWS}: (1) ALBERT is improved by {28.3%, 15.7%, 20.6%, 27.2%} (2) DistilBERT is improved by {16.1%, 15.1%, 4.3%, 14.0%} (3) RoBERTa is improved by {11.4%, 1.8%, 4.3%, 5.4%} (4) BERT is improved by {19.5%, 16.6%, 14.3%, 13.1%}.

- Our method, Pro-BERT (MCP) + AT, exhibits best robustness among all the baselines. By simply plugging in ProAttention (MCP) module without fine-tuning, our Pro-BERT can achieve comparable robustness to most adversarial training-based methods which require substantial computational time and resources. Furthermore, our framework is orthogonal to most existing defenses, allowing for combined use with them to further enhance robustness. For instance, when combined with AT technique, our Pro-BERT (MCP) + AT can further improve BERT + AT by {14.7%, 4.6%, 18.6%, 6.2%} under {TextFooler, TextBugger, DeepWordBug, PWWS}.

Ablation Study. To further validate the effectiveness of our ProTransformer, we provide comprehensive ablation studies on *Convergence, Adversarial fine-tuning, Attack constraints, Different penalties, Different backbones,* and *Running time* in Appendix D.2.2.

3.2 Adversarial Prompting Attacks on LLMs

In the context of prompt-based generative AI, the adversarial attacks mechanisms on LLMs become more enriched and sophisticated. In this section, we will evaluate the robustness of our proposed ProTransformer under two popular attacks: **prompt attack** and **jailbreak**.

3.2.1 Prompt Attack

We display the results of T5 and LLaMA in Figure 3 and Appendix N.2. Due to the space limit, we leave the detailed experiments and discussions in Appendix D.3.1. From the figure, we can observe that our Pro-T5 (MCP) exhibits a significant advantage over other methods, and this advantage becomes even more evident as the number of perturbed words increases.

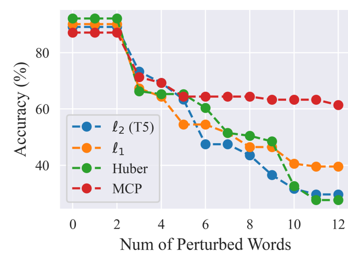


Figure 3: Prompt attack of T5.

3.2.2 Jailbreak Attack

For jailbreaking, we select GCG (Zou et al., 2023) method to conduct suffix-injection attacks to evaluate the resilience of models comprehensively. The detailed results and analysis are available in Appendix D.3.2.

The comparison of our Pro-Vicuna and backbone Vicuna under adaptive jailbreak is presented in Table 5. We can find that our Pro-Vicuna can significantly improve Vicuna by an average of 10.4% across various numbers of attack queries.

Table 2: ASRs under adaptive jailbreak.

Num of Queries	12	13	14	15	16	17	18	19	20
Vicuna	61.4	65.2	71.5	75.8	78.7	82.6	84.1	86.5	87.4
Pro-Vicuna (Huber)	50.7	55.9	60.8	64.3	67.4	70.5	74.0	77.7	78.6

4 Experiment beyond Language Modeling

Beyond language domain, as shown in Figure 2, our ProAttention is a fundamental module which can reinforce any attention-based models across various domains or modalities. We integrate ProAttention into **vision** models and **graph** learning models to further validate the effectiveness and generalization of our approach in Appendix E.

5 Conclusion

In this paper, we delve into the robustness and security of the popular transformer-based architectures. We revisit the vulnerability of attention mechanisms with theoretical understanding and simulations. We propose an interpretable robust attention layer to robustify any transformer architecture via a plug-and-play paradigm. Our proposed ProAttention is an effective, efficient, and universal framework that can significantly enhance the robustness of transformers across various tasks, architectures, attacks, and domains with only 4 lines of code but without additional training or fine-tuning. We are excited about the future of the proposed ProTransformer architecture and hope to see its full potential with training or fine-tuning on large models in the future.

REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=p-BhZSz59o4`.

Roy Bar-Haim, Ido Dagan, and Idan Szpektor. Benchmarking applied semantic inference: The pascal recognising textual entailment challenges. In *Language, Culture, Computation. Computing-Theory and Technology: Essays Dedicated to Yaacov Choueka on the Occasion of His 75th Birthday, Part I*, pp. 409–424. Springer, 2014.

Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. Universal adversarial attacks on text classifiers. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7345–7349. IEEE, 2019.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8, 2009.

Peter Bloomfield and William L Steiger. *Least absolute deviations: theory, applications, and algorithms*, volume 6. Springer, 1983.

Hezekiah J Branch, Jonathan Rodriguez Cefalu, Jeremy McHugh, Leyla Hujer, Aditya Bahl, Daniel del Castillo Iglesias, Ron Heichman, and Ramesh Darwishi. Evaluating the susceptibility of pre-trained language models via handcrafted adversarial examples. *arXiv preprint arXiv:2209.02128*, 2022.

Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The annals of applied statistics*, 5(1):232, 2011.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(1):105–105, 2010.

Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Gunturk. Iteratively re-weighted least squares minimization for sparse recovery, 2008.

Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. Jailbreaker: Automated jailbreak across multiple large language model chatbots. *arXiv preprint arXiv:2307.08715*, 2023.

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

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions. *arXiv preprint arXiv:2107.13541*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pp. 2286–2296. PMLR, 2021.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, 2018.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1): 17–40, 1976.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56. IEEE, 2018.

Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6174–6181, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.498. URL https://aclanthology.org/2020.emnlp-main.498.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34:7637–7649, 2021.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

Yotam Gil, Yoav Chai, O. A. Gorodissky, and Jonathan Berant. White-to-black: Efficient distillation of black-box adversarial attacks. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:102353948.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

Roland Glowinski and Americo Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Xing Han, Tongzheng Ren, Tan Minh Nguyen, Khai Nguyen, Joydeep Ghosh, and Nhat Ho. Designing robust transformers using robust kernel density estimation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=BqTv1Mtuhu.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving verified robustness to symbol substitutions via interval bound propagation. *arXiv preprint arXiv:1909.01492*, 2019.

Peter J Huber. Robust regression: asymptotics, conjectures and monte carlo. *The annals of statistics*, pp. 799–821, 1973.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1875–1885, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1170. URL https://aclanthology.org/N18-1170.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4129–4142, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1423. URL https://aclanthology.org/D19-1423.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020a.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 66–74, 2020b.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatGPT. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=ls4Pfsl2jZ.

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

Linyang Li and Xipeng Qiu. Token-aware virtual adversarial training in natural language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8410–8418, 2021.

Naihan Li, Yanqing Liu, Yu Wu, Shujie Liu, Sheng Zhao, and Ming Liu. Robutrans: A robust transformer-based text-to-speech model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8228–8235, 2020.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. Searching for an effective defender: Benchmarking defense against adversarial word substitution, 2021.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

Junhua Liu, Trisha Singhal, Lucienne TM Blessing, Kristin L Wood, and Kwan Hui Lim. Crisisbert: a robust transformer for crisis classification and contextual crisis embedding. In *Proceedings of the 32nd ACM conference on hypertext and social media*, pp. 133–141, 2021a.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023a.

Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. Jailbreaking chatgpt via prompt engineering: An empirical study. *arXiv preprint arXiv:2305.13860*, 2023b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021b.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJzIBfZAb`.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 119–126, 2020.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pp. 49–54. IEEE, 2016.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1085–1097, 2019.

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. Smoothllm: Defending large language models against jailbreaking attacks, 2023.

Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. Interpretable adversarial perturbation in input embedding space for text. In *27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pp. 4323–4330. International Joint Conferences on Artificial Intelligence, 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. Better robustness by more coverage: Adversarial training with mixup augmentation for robust fine-tuning. In *Findings of ACL*, 2021.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1170`.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.

Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. Infobert: Improving robustness of language models from an information theoretic perspective. In *International Conference on Learning Representations*, 2021.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jA235JGM09.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4816–4823. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/669. URL https://doi.org/10.24963/ijcai.2019/669.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective, 2019.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. TableFormer: Robust transformer modeling for table-text encoding. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 528–537, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.40. URL https://aclanthology.org/2022.acl-long.40.

Mao Ye, Chengyue Gong, and Qiang Liu. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3465–3475, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.317. URL https://aclanthology.org/2020.acl-main.317.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

Jiehang Zeng, Jianhan Xu, Xiaoqing Zheng, and Xuanjing Huang. Certified robustness to text adversarial attacks by randomized [mask]. *Computational Linguistics*, 49(2):395–427, 2023.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. 2010.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems*, 33:9263–9275, 2020.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

Yiming Zhang and Daphne Ippolito. Prompts should not be seen as secrets: Systematically measuring prompt extraction attack success. *arXiv preprint arXiv:2307.06865*, 2023.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huan. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In *Association for Computational Linguistics (ACL)*, 2021.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BygzbyHFvB`.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1399–1407, 2019.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. Promptbench: A unified library for evaluation of large language models, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A Pseudocode of Plug-and-Play Robust Attention (ProAttention)

Here, we provide the complete pseudocode of our ProAttention including various penalties cases in Algorithm 2. The core itertions are show in the for loop in the code. Our ProAttention is easy to implement by only replacing the vanilla attention module with our ProAttention.

Algorithm 2 ProAttention in PyTorch style

```python
class ProAttention(nn.Module):
    def __init__(self, K, gamma, delta, penalty):
        super().__init__()
        self.K = K
        self.gamma = gamma
        self.delta = delta
        self.penalty = penalty

    def forward(self, A, V):
        Z = torch.matmul(A, V) # Initialization

        if self.penalty == 'L2':
            return Z # Original attention

        for _ in range(self.K):
            D = torch.cdist(Z, V) # Pairwise distance

            if self.penalty == 'L1':
                W = 1/D
            elif self.penalty == 'MCP':
                W = torch.clip(1/D - 1/self.gamma, min=0)
            elif self.penalty == 'Huber':
                W = torch.clip(self.delta/D, max=1)
            elif self.penalty == 'Huber-MCP':
                W = torch.clip(self.delta/(self.gamma-self.delta)*(self.gamma/D-1), min=0, max=1)

            W = torch.nn.functional.normalize(W * A, p=1, dim=-1) # Normalization
            Z = torch.matmul(W, V) # Update
        return Z
```

B ProTransformer

In this section, we interpret the self-attention mechanism in transformers as the weighted least square estimator and propose robust alternatives. An efficient Newton-IRLS algorithm is proposed to be unrolled as the plug-and-play robust attention layer to robustify any transformer architecture.

B.1 Attention Mechanism as WLS Estimator

In this work, we aim to design a robust transformer architecture without additional training or fine-tuning. To this end, we first establish a simple but intriguing connection between the attention mechanism and the weighted least squares (WLS) estimator. Specifically, the classical scaled dot-product attention can be formulated as:

$$\operatorname{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V} := \mathbf{AV} \tag{3}$$

where $\mathbf{Q} = \{\boldsymbol{q}_j\}_{j\in[N]}, \mathbf{K} = \{\boldsymbol{k}_j\}_{j\in[N]}, \mathbf{V} = \{\boldsymbol{v}_j\}_{j\in[N]}$ are the query, key and value matrices, respectively. $\mathbf{A} = \{a_{ij}\}_{i,j\in[N]}$ is the attention matrix, and d_k is the dimension of queries and keys. For every token $i \in [N]$, the aggregated output embedding is essentially the weighted sum over the embedding (value vectors) of input tokens: $\sum_{j=1}^N a_{i,j}\boldsymbol{v}_j$, which can be rewritten as $\sum_{j=1}^N a_j\boldsymbol{v}_j$ by omitting the index i for simplicity. Therefore, it is natural to interpret the weighted sum as the optimal solution of the following weighted least squares (WLS) error minimization problem by considering its first-order optimality condition:

$$\arg\min_{\boldsymbol{z}} \sum_{j=1}^N a_j \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2. \tag{4}$$

The square loss (ℓ_2 or $\|\cdot\|^2$) in Eq. (4) causes larger residuals to exert a disproportionately higher influence on the final estimation. In this way, the substantial residuals created by the outliers tend to dominate the objective to be minimized. As a result, the WLS estimator may excessively adjust the estimator to accommodate these outliers, leading to a heavily biased representation of the majority of the data. To better illustrate the estimation bias of WLS estimator, we simulate a mean estimation problem using synthetic data. The detailed setting of numerical simulation is provided in the Appendix L. As shown in Figure 5, as the ratio of outliers grows, the ℓ_2-based WLS estimator deviates increasingly from the true mean due to the estimation bias accumulated by the outlying data points. In the extreme case, even one single outlier can enforce the WLS estimator to be any target solution by an adversarially chosen outlier.

B.2 Robust Estimators

The analysis and simulated experiments presented above clearly demonstrate that the WLS estimator suffers from a catastrophic bias effect when facing outliers. Similarly, the attention mechanism, which essentially operates as the WLS estimator, is highly sensitive to outliers. This perspective provides a valid explanation of why various attention-based transformer architectures are easily manipulated and compromised by introducing adversarial noises in the input data.

Our interpretation of the attention mechanism in transformers as WLS estimator provides a rigorous perspective to design robust alternatives. To dampen the effect of the outliers, multiple robust regression algorithms have been proposed in robust statistics using least absolute deviations Bloomfield & Steiger (1983), Huber regression Huber (1973), and Minimax Concave Penalty (MCP) Zhang (2010). Motivated by these advancements with rigorous robustness guarantees, we propose the robust weighted sum estimators as follows:

$$\arg\min_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{z}) = \sum_{j=1}^N a_j \cdot \rho(\|\boldsymbol{v}_j - \boldsymbol{z}\|) \tag{5}$$

Special cases of ρ. Specifically, the square loss $\rho(z) = \frac{1}{2}z^2$ recovers the WLS estimator while $\rho(z) = z$ results in the Least Absolute Deviation (LAD) estimator. Additionally, we also consider Huber loss $\rho_\delta(z)$ and MCP function $\rho_\gamma(z)$. Moreover, we propose the combination Huber-MCP function $\rho_{\delta,\gamma}(z)$ which integrates the advantages of Huber and MCP. We plot these potential penalties in Figure 4 and provide their detailed formulations in Appendix H.

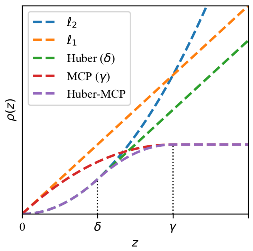


Figure 4: Different penalties $\rho(z)$.

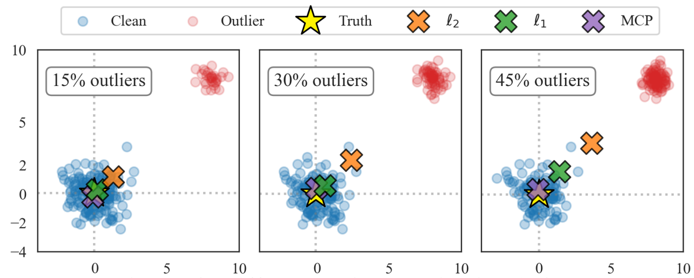


Figure 5: Different estimators in simulations.

From Figure 4, we can make the following observations: (1) ℓ_2 loss increase quadratically with z, which is much faster than the linear increase of ℓ_1 loss. This suggests that ℓ_2 loss is more sensitive to the residual magnitude than ℓ_1 loss; (2) Huber loss is equivalent to the ℓ_2 loss within the range $(0, \delta)$, and it becomes similar to ℓ_1 when $z > \delta$, which indicates that Huber loss may mitigate the effect of large noise while keeping decent performance in noiseless scenario; (3) MCP loss becomes constant when z is large, and this transition can be adjusted by the thresholding parameter γ. Particularly, when γ approaches infinity, the penalty $\rho_\gamma(z)$ reduces to the ℓ_1 case; (4) We propose Huber-MCP $\rho_{\delta,\gamma}(z)$ as a combination of Huber and MCP which shares their advantages in both low- and high-value regions.

B.3 Newton-IRLS Algorithm

The proposed robust estimator in Eq. (5) is non-convex and non-smooth, posing a challenge for efficient algorithm design. Moreover, the exploding model size of evolving transformers further necessitates the design of efficient neural network layers. To this end, we propose a novel Newton Iteratively Reweighted Least Squares (Newton-IRLS) algorithm to approximate the solution of Eq. (5) as follows:

$$\boldsymbol{z}^{(k+1)} = \frac{\sum_j a_j \cdot w_j^{(k)} \cdot \boldsymbol{v}_j}{\sum_j a_j \cdot w_j^{(k)}} \tag{6}$$

where $w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}$ and ρ' is the derivative of ρ. Its convergence is guaranteed by Theorem F.2 in Appendix F. We leave the detailed motivation and algorithm design in Appendix F and theoretical proof in Appendix G.

Interpretability. The Newton-IRLS algorithm in Eq. (6) can be interpreted as a reweighted sum, in which the derived $w_j^{(k)}$ modifies the original attention score a_j on the value vector $\boldsymbol{v}_j$. By choosing different penalty functions ρ on the residuals $\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|$, we obtain various reweighting schemes ($w_j^{(k)}$) as follows: (1) ℓ_2 loss: $w_j^{(k)} = 1$; (2) ℓ_1 loss: $w_j^{(k)} = \frac{1}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}$; (3) Huber loss: $w_j^{(k)} = \min\left[1, \frac{\delta}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}\right]$; (4) MCP loss: $w_j^{(k)} = \max\left[\frac{1}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} - \frac{1}{\gamma}, 0\right]$; (5) Huber-MCP loss: $w_j^{(k)} = \max\left[\min\left[\frac{\delta}{\gamma-\delta}\left(\frac{\gamma}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} - 1\right), 1\right], 0\right]$.

Specifically, ℓ_2 loss can recover the vanilla attention since the weights are 1. With ℓ_1 loss, the weight is inversely proportional to $\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|$, which up-weights the inliers and down-weights the outliers. Huber loss behaves as the ℓ_2 loss when $\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\| < \delta$ and resembles ℓ_1 loss for $\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\| > \delta$. The weight derived by MCP loss enhances the interpretability of the robust estimator by down-weighting or completely removing the outliers. The weight w_j becomes smaller as the distance $\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|$ increases, thereby down-weighting the outlying cases. When this distance exceeds the threshold γ, the weight becomes 0, totally removing the outliers. In the case of Huber-MCP loss, the weight combine the advantages of recovering the ℓ_2 loss and largely mitigating the outliers. The detailed derivation of all the cases can be found in Appendix H.

Simulation. To better illustrate the robustness of our proposed estimators, we visualize different estimators under various outlier ratios on the synthetic data as shown in Figure 5. The detailed experiment settings are provided in Appendix L.2. We can observe that MCP $> \ell_1 > \ell_2$ in terms of estimation robustness, and the robustness becomes increasingly significant as the number of outliers grows.

B.4 PLUG-AND-PLAY ROBUST ATTENTION LAYER

In the previous subsection, we formulate the token-wise Newton-IRLS approach for notation simplicity. Here, we will present the corresponding matrix version for the entire attention layer, which can be implemented efficiently.

Matrix Form. Let $\mathbf{Z}^{(k)} = \{\boldsymbol{z}_i^{(k)}\}_{i\in[N]}$ be the estimator for token i at the k-th iteration. Subsequently, the pairwise distance $\mathbf{D}^{(k)} = \{\|\boldsymbol{v}_j - \boldsymbol{z}_i^{(k)}\|\}_{i,j\in[N]}$ between $\mathbf{Z}^{(k)}$ and $\mathbf{V}$ can be simply and efficiently computed using the `torch.cdist` function in PyTorch. Following this, the weight $\mathbf{W}^{(k)} = \{w_{ij}^{(k)}\}_{i,j\in[N]}$ can be calculated element-wise based on $\mathbf{D}^{(k)}$. Then the next step $\mathbf{Z}^{(k+1)}$ is updated as a reweighted matrix multiplication $(\mathbf{W}^{(k)} \odot \mathbf{A}) \cdot \mathbf{V}$.

Plug-and-Play Robust Attention. Instead of focusing on interpreting the attention scores $\{a_j\}_{j\in[N]}$, we consider the attention scores as given variables. Therefore, the proposed algorithm can be packaged as a robust attention module, which can be readily plugged into the transformers as a **P**lug-and-Play **Ro**bust Attention (**ProAttention**) layer without additional training or fine-tuning as shown in Figure 2. The implementation of ProAttention using MCP penalty in PyTorch is shown in Algorithm 3. The complete pseudocode for other penalties is presented in in Appendix A.

Algorithm 3 ProAttention (MCP)

```
D = torch.cdist(Z, V) # Pairwise distance
W = torch.clip(1/D-1/gamma, min=0) # MCP
W = normalize(W * A, p=1, dim=-1)
Z = torch.matmul(W, V) # Update
```

Complexity analysis. Let N, D, and K represent the length of tokens, the dimension of vectors, and the steps of the iterations, respectively. The vanilla attention requires $2 \cdot N \times N \times D$ basic operations while our ProAttention needs $(1 + 2K) \cdot N \times N \times D$. However, our ProAttention remains efficient, as the Newton-IRLS method can effectively approximate the solution within only 3 steps ($K \leq 3$) and ProTransformers do not introduce additional computation for training or fine-tuning. We provide the detailed complexity analysis of various attentions in Appendix R.

Advantages. Our proposed ProAttention enjoys the following advantages: (1) *Simplicity*: it is simple and easy to implement with only 4 core lines of code in Algorithm 3; (2) *Efficiency*: it is a plug-and-play layer that can be integrated into any trained transformer without additional training or fine-tuning; (3) *Universility*: it is a universal framework that advances the vanilla attention mechanism into a series of robust derivatives with different penalties. Moreover, it can be applied to any attention-based model across various modalities and tasks.

In the following sections, we will present comprehensive experiments and studies to validate the effectiveness of the proposed ProAttention on language modeling in Section D as well as computer vision and graph learning in Section E.

C Related Work

In this section, we summarize related works on the attack and defense of transformers focusing on language domains.

Attacks. Compared to the well-established attack mechanisms in vision domain Goodfellow et al. (2014); Madry et al. (2018), the text attacks in language domain are highly complicated due to the natural irregularity of data structure. According to the perturbation units, text attacks can be classified into character-level Gao et al. (2018); Gil et al. (2019), word-level Papernot et al. (2016); Samanta & Mehta (2017); Sato et al. (2018); Behjati et al. (2019); Ren et al. (2019); Jin et al. (2020a); Garg & Ramakrishnan (2020), sentence-level Iyyer et al. (2018), and multi-level Liang et al. (2017); Ebrahimi et al. (2018); Li et al. (2018). These classic text attacks typically generate adversarial examples through misspellings, synonym replacement, etc. In the era of LLMs, several new types of attacks have emerged, such as jailbreak attacks Wei et al. (2023); Li et al. (2023); Deng et al. (2023); Liu et al. (2023b) and prompt injection Branch et al. (2022); Zhang & Ippolito (2023); Liu et al. (2023a). These prompting-based attacks aim to trick models into generating unsafe outputs using adversarially crafted prompts.

Defenses. There have been some works proposed to defend against adversarial text attacks, which can be roughly divide into empirical defenses Si et al. (2021); Li & Qiu (2021); Wang et al. (2021); Dong et al. (2021); Zhou et al. (2021) and certifiable defenses Ye et al. (2020); Zeng et al. (2023); Huang et al. (2019); Jia et al. (2019). These defenses require excessive computation costs for training, inference, or both. Nevertheless, all these methods are typically architecture-agnostic, which are orthogonal to and can be combined with our proposed defenses on the transformer architecture to further enhance the robustness.

To safeguard the transformers, several endeavors have been made from the transformer architecture perspective. Li et al. (2020) modify the attention mechanism and position embedding to robustify text-to-speech transformers. In the crisis detection and recognition task, Liu et al. (2021a) propose an end-to-end attention-based classifier to enhance robustness. For tabular data, TableFormer Yang et al. (2022) adopts structural-aware table-text encodings, and is robust to row and column order perturbations. However, these architectures are tailored for specific tasks, which require specific domain knowledge and are hard to generalize across tasks. Han et al. (2023) propose a general framework for self-attention modules via robust kernel density estimation (RKDE). However, this method introduces excess computation cost and shows relatively limited robustness improvement. Generally speaking, existing approaches either require substantial computations or rely on specific domain knowledge, which hinders their extensions to larger models or broader application domains.

D MAIN EXPERIMENT ON LANGUAGE MODELING

In this section, we evaluate the effectiveness of the proposed Plug-and-play Robust Transformers (ProTransformers) under classic text attacks on pre-trained language models, and two prompting-based attacks (prompt attack and jailbreak attack) in the context of LLMs. We also provide comprehensive ablation studies.

D.1 EXPERIMENT SETTING

Tasks and Datasets. For topic classification, we use AG's News Corpus (AGNEWS) Zhang et al. (2015). For sentiment analysis, we utilize two widely-used datasets: Internet Movie Database (IMDB) Maas et al. (2011) and Stanford Sentiment Treebank (SST-2) Socher et al. (2013). For textual entailment, we make use of Recognizing Textual Entailment (RTE) in the General Language Understanding Evaluation benchmark Wang et al. (2019). For jailbreak attack, we select a new dataset Behaviors introduced in Zou et al. (2023). For the detailed information on these datasets, please refer to Appendix I.

Backbone Architectures. For classical pre-trained language models, we choose BERT Devlin et al. (2018) and its variants including RoBERTa Liu et al. (2019), ALBERT Lan et al. (2020) and DistilBERT Sanh et al. (2019). For large language models (LLMs), we choose T5 Raffel et al. (2023), LLaMA Touvron et al. (2023) and Vicuna Chiang et al. (2023). For the detailed information on backbone architectures, please refer to Appendix J.2.

Attacks. We not only evaluate several classic text attacks but also include popular prompt attacks and jailbreak attacks on the LLMs. The three attack mechanisms and their differences are illustrated in Figure 1. For classic text attacks, we evaluate the attacks at various levels, including the character-level DeepWordBug Gao et al. (2018), word-level PWWS Ren et al. (2019), TextFooler Jin et al. (2020a), and multi-level TextBugger Li et al. (2018). For prompt attacks, we modify the prompt template according to the aforementioned text attacks following the evaluation setting in PromptBench Zhu et al. (2024). For jailbreak, we evaluate the suffix attack using Greedy Coordinate Gradient (GCG) method Zou et al. (2023) and we test both attacks transferred from surrogate model Vicuna (transfer attack) and attacks directly targeting the victim models (adaptive attack). Please refer to Appendix K for details on attacks.

Defense Baselines. We include the following defense baselines in our experiments: MixADA Si et al. (2021), PGD-Adv Madry et al. (2018), FreeLB Zhu et al. (2020), TA-VAT Li & Qiu (2021) and SmoothLLM Robey et al. (2023). Additionally, we also include the adversarial training (AT), wherein the augmented perturbations are generated by the attack to be assessed. Details of these defense methods are provided in Appendix J.1.

Evaluation metrics. Following Li et al. (2021), we use 3 metrics to evaluate the model performance. Clean accuracy (**Clean%**) is the model accuracy on the clean testing data. Accuracy under attack (**AUA%**) is the model accuracy on the perturbed data under specific attack. Attack success rate (**ASR%**) is the ratio of the number of successfully perturbed cases divided by the number of attempted texts.

Hyperparameters. For text attack setting, we follow the setting in the TextAttack Morris et al. (2020) framework. For prompt attack, we follow the setting in PromptBench Zhu et al. (2024). For GCG-based jailbreak attack, we follow the setting in Robey et al. (2023). The detailed attack settings can be found in Appendix K. For defense baselines, we follow the settings in their original papers. For our ProTransformer, we set the default number of ProAttention layers as $K = 3$ since our algorithm can efficiently converge within 3 steps. Then we tune the δ (default 1) or γ (default 4) in the penalties to obtain the optimal parameters.

In this section, we evaluate the effectiveness of the proposed Plug-and-play Robust Transformers (ProTransformers) under classic text attacks on pre-trained language models, and two prompting-based attacks (prompt attack and jailbreak attack) in the context of LLMs. We also provide comprehensive ablation studies. Following Li et al. (2021), we use 3 metrics to evaluate the model performance: (1) Clean accuracy (**Clean%**), accuracy under attack (**AUA%**) and attack success rate (**ASR%**). The complete experimental setting information can be found in Appendix D.1.

D.2 Classic Text Attacks on Language Models

D.2.1 Adversarial Robustness

Table 3: The results of topic classification on AGNEWS.

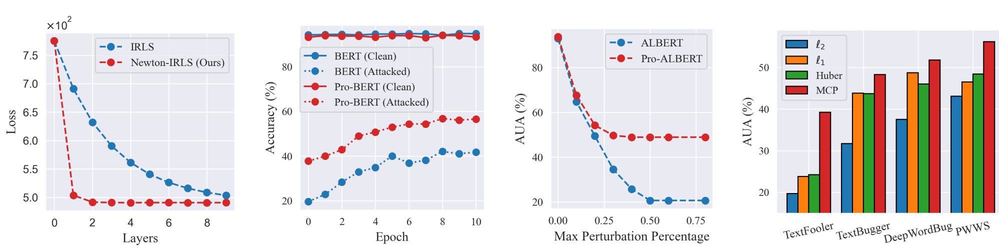
		Textfooler		TextBugger		DeepWordBug		PWWS	
Model	Clean% ↑	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓	Aua% ↑	ASR% ↓
ALBERT	93.0	20.6	77.9	26.1	71.9	38.9	58.2	35.9	61.4
Pro-ALBERT (MCP) (Ours)	93.8	48.9	47.3	41.8	55.3	59.5	35.9	63.1	32.0
DistilBERT	93.5	13.2	85.9	33.6	63.4	30.0	67.9	36.5	61.0
Pro-DistilBERT (MCP) (Ours)	93.9	29.3	68.5	48.7	47.9	34.3	63.1	50.5	45.6
RoBERTa	93.4	13.0	86.1	32.5	64.5	41.2	55.9	34.0	63.6
Pro-RoBERTa (MCP) (Ours)	93.7	24.4	73.7	34.3	62.8	45.5	51.5	39.4	57.5
BERT	94.2	19.7	78.9	31.7	67.5	37.5	59.8	43.1	53.8
+ FreeLB	94.2	38.0	59.5	42.8	55.5	56.1	40.9	57.0	39.9
+ PGD	94.1	36.8	61.7	40.5	57.1	47.6	49.7	48.7	48.6
+ MixADA	94.3	35.6	62.4	35.4	62.9	38.2	50.5	46.8	50.4
+ TA-VAT	**94.4**	36.2	61.8	39.2	58.2	49.5	48.1	47.0	50.7
+ AT	94.1	42.1	54.8	56.1	39.4	42.4	54.1	62.6	32.5
Pro-BERT (ℓ_1) (Ours)	94.2	23.8	74.5	43.8	53.0	48.7	47.8	46.5	50.1
Pro-BERT (Huber) (Ours)	94.2	24.2	74.0	43.7	52.9	46.0	50.5	48.4	47.9
Pro-BERT (MCP) (Ours)	93.2	39.2	57.7	48.3	48.5	51.8	43.8	56.2	39.2
Pro-BERT (MCP) + AT (Ours)	94.0	**56.8**	**38.9**	**60.7**	**35.1**	**61.0**	**34.1**	**68.8**	**25.7**

To demonstrate the effectiveness of the proposed ProTransformer, we compare the robustness of our methods with several popular defenses in three classical tasks: topic classification, sentiment analysis, and textual entailment.

Performance analysis. The experimental results of topic classification (AGNEWS) are presented in Table 3, and we provide the results of sentiment analysis (IMDB) and textual entailment (RTE) in Appendix M.1 and M.2 due to the space limit. From the experiment results, we can make the following observations:

- Our ProAttention is a highly effective plug-in module that significantly and consistently enhances the robustness of various transformer-based backbones across all kinds of adversarial attacks. Taking AGNEWS as the instance, when combined with our ProAttention (MCP), under the attacks {Textfooler, TextBugger, DeepWordBug, PWWS}: (1) ALBERT is improved by {28.3%, 15.7%, 20.6%, 27.2%} (2) DistilBERT is improved by {16.1%, 15.1%, 4.3%, 14.0%} (3) RoBERTa is improved by {11.4%, 1.8%, 4.3%, 5.4%} (4) BERT is improved by {19.5%, 16.6%, 14.3%, 13.1%}.
- Our method, Pro-BERT (MCP) + AT, exhibits best robustness among all the baselines. By simply plugging in ProAttention (MCP) module without fine-tuning, our Pro-BERT can achieve comparable robustness to most adversarial training-based methods which require substantial computational time and resources. Furthermore, our framework is orthogonal to most existing defenses, allowing for combined use with them to further enhance robustness. For instance, when combined with AT technique, our Pro-BERT (MCP) + AT can further improve BERT + AT by {14.7%, 4.6%, 18.6%, 6.2%} under {TextFooler, TextBugger, DeepWordBug, PWWS}.

D.2.2 Ablation Study



(a) Algorithm Convergence (b) Adversarial Fine-tuning (c) Attack Constraints (d) Different Penalties

Figure 6: Ablation studies.

Convergence. To validate the advantage of our Newton-IRLS over the first-order method, we conduct a simulation experiment and plot the loss descent curves in Figure 6 (a). It can be observed that our Newton-IRLS exhibits efficient convergence as claimed in Section B.3. We provide the experiment details, all the loss descent curves (Figure 9) as well as the visualization of trajectories (Figure 10) of the updated vectors in 2D plane in Appendix L.1 to further demonstrate the effectiveness of our algorithm.

Adversarial fine-tuning. To get insight into how the models gain more robustness from adversarial examples, we track the training curves of adversarial fine-tuning under TextFooler in Figure 6 (b), and put the results of other attacks in Figure 11 in Appendix M.3. We can observe that our Pro-BERT (MCP) is compatible with adversarial fine-tuning technique to further enhance the model resilience.

Attack constraints. In text attack, there are several kinds of attakc constraints including the maximum percentage of perturbed words, minimum cosine similarity between the replaced synonym and original word, and minimum sentence similarity threshold between the original sentence and perturbed sentence. We test the values of these constraints in TextFooler. We present the results under different perturbation percentages in Figure 6 (c) and other constraint measurements in Appendix M.4. From the results, we observe that our Pro-ALBERT (MCP) can significantly outperform the backbone ALBERT across all ranges of constraints.

Different penalties. Our Newton-IRLS is flexible to be formulated as different robust estimators with different penalties. From the comparison in Figure 6 (d) , it can be observed that our robust framework can consistently improve the robustness of the backbone BERT (ℓ_2). Specifically, ℓ_1 and Huber-based defenses are comparable, and MCP-based method exhibits the best performance among them.

Different backbones. Our method is a universal plug-and-play layer applicable to various attention-based backbones. The results in Table 3 and the ablation study on different backbones in Appendix M.5 (Figure 13) demonstrate that our ProAttention can improve the robustness over all kinds of architecture backbones (BERT, RoBERTa, DistilBERT and ALBERT) against various attacks with significant margins.

Running time. To empirically evaluate the efficiency of our method, we test the average running time on AGNEWS using BERT and Pro-BERT (MCP) equipped with multi-layer (K) ProAttention. The results in Table 4 show that our ProAttention only requires 1-2 times additional inference time of the backbone model yet achieves significant improvement in robustness without training.

Table 4: Average running time (ms) on AGNEWS.

	BERT	Pro-BERT (MCP)					
Running time (ms)	6.14	9.04	11.67	14.34	17.33	19.89	21.87
# Layers (K)	\	1	2	3	4	5	6

D.3 Adversarial Prompting Attacks on LLMs

In the context of prompt-based generative AI, the adversarial attacks mechanisms on LLMs become more enriched and sophisticated. In this section, we will evaluate the robustness of our proposed ProTransformer under two popular attacks: **prompt attack** and **jailbreak**.

D.3.1 Prompt Attack

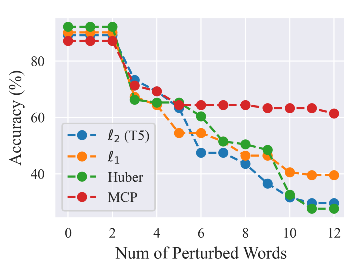


Figure 7: Prompt attack of T5.

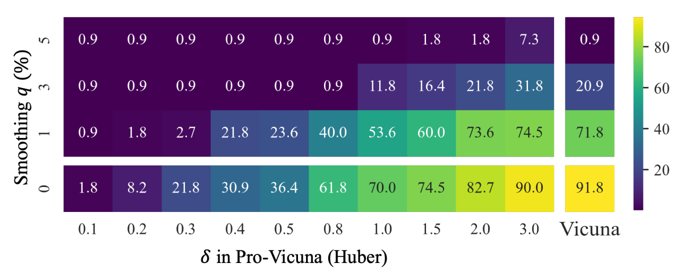


Figure 8: Attack success rates (ASRs) under transfer jailbreak.

As shown in Figure 1, the most significant distinction between prompt attacks and classical text attacks is that prompt attacks aim to mislead the models by altering the prompt template rather than the input content. We display the results of T5 in Figure 7 and leave the comprehensive study in Appendix N.1. We also present the results on LLaMA in Appendix N.2. From the results, we can make the following observations: (1) For T5, the choice of the penalty would affect the robustness of defenses. Specifically, Pro-T5 (MCP) exhibits a significant advantage over other methods, and this advantage becomes even more evident as the number of perturbed words increases. Pro-T5 (ℓ_1) and Pro-T5 (Huber) show a slight improvement over the backbone model T5. (2) For LLaMA, Huber-MCP and Huber-based methods exhibit better robustness than other methods while preserving good clean performance. The detailed experiments and discussions can be found in Appendix N.2.

D.3.2 Jailbreak Attack

In recent years, prompts have played a pivotal role in guiding models to generate desired outputs. Nevertheless, there exist malicious "jailbreak prompts", which are intentionally designed to bypass the built-in safeguards in LLMs, causing the model to produce harmful content that violates the legal policies. As illustrated in Figure 1, the suffix-injection jailbreaks attempt to append a non-semantic suffix to the user's prompt to fool the models. We select GCG method under both transfer and adaptive attack settings to evaluate the resilience of models comprehensively.

Transfer Jailbreak. In Figure 8, we compare the Attack Success Rates (ASRs) of Vicuna and its corresponding Pro-Vicuna (Huber) with various δ values on Behaviors. In each column, we also include SmoothLLM Robey et al. (2023) with different smoothing extent $q(\%)$ to further reinforce the resilience of every single model. The last row of matrix ($q = 0$) stands for the performance without random smoothing. The additional results of random smoothing with swap, insert and patch are presented in Appendix O.

From the results, we can observe that: (1) Our Pro-Vicuna can significantly improve the robustness of Vicuna. As shown in the last row of Figure 8, with $\delta = 0.1$, we successfully reduce the ASR to 1.8%, which is comparable to the random smoothing defense that requires multiple random perturbations, inferences and aggregations. (2) Our ProAttention is orthogonal to randomized smoothing defense and can be combined with it to further improve the robustness.

Table 5: Attack success rates (ASRs) under adaptive jailbreak.

Num of Queries	12	13	14	15	16	17	18	19	20
Vicuna	61.4	65.2	71.5	75.8	78.7	82.6	84.1	86.5	87.4
Pro-Vicuna (Huber)	50.7	55.9	60.8	64.3	67.4	70.5	74.0	77.7	78.6

Adaptive Jailbreak. Although our Pro-Vicuna has demonstrated significant effectiveness under transfer attack (black-box), it is still unclear whether our method can be resilient under white-box attacks which adaptively target the specific victim models. The comparison of our Pro-Vicuna and backbone Vicuna under adaptive jailbreak is presented in Table 5 and the additional experiment results are available in Appendix O. Our Pro-Vicuna can improve Vicuna by an average of 10.4% across various numbers of attack queries. We don't include SmoothLLM in adaptive attacks since it introduces non-differentiable operators that preclude the gradient-based GCG attack on it.

E Experiment beyond Language Modeling

In previous section, we have provided comprehensive experiments to validate the effectiveness of our ProTransformer in the (large) language models. In fact, as shown in Figure 2, our ProAttention is a fundamental module which can reinforce any attention-based models across various domains or modalities. In this section, we will integrate ProAttention into vision models and graph learning models to further validate the effectiveness and generalization of our approach.

E.1 Image Classification

In vision domain, we conduct two attacks (FGSM and PGD) on several vision transformers including ViT, BeiT, ConviT, DeiT and Swin. We perform the experiments on CIFAR-10 across budgets $\{\frac{1}{255}, \frac{4}{255}, \frac{8}{255}\}$, and present the results of PGD in Table 6 and additional experiments in Appendix P. From the results, we can observe that our Pro-ViT (MCP) can outperform other baselines with a significant margin, indicating that MCP loss show excellent capability to mitigate the effect of the outliers. Significantly, Pro-ViT (MCP) can outperform the second best model by $\{31.93\%, 44.92\%, 35.64\%\}$ under different budgets.

Table 6: Adversarial robustness on CIFAR-10 (PGD).

Model\Budget	0 (Clean)	$1/255$	$4/255$	$8/255$
DeiT	97.91	<u>38.98</u>	0.44	0.0
ConViT	98.70	<u>41.75</u>	1.83	0.0
BeiT	97.87	6.81	0.0	0.0
Swin	98.30	14.89	0.02	0.01
ViT	<u>98.74</u>	34.61	1.83	0.26
Pro-ViT (MCP) (Ours)	98.40	**77.39**	**48.11**	**33.40**

E.2 Graph Representation Learning

Besides the language and vision domains, we also validate the effectiveness of our method in the graph domain. We conduct the semi-supervised node classification task and leverage PGD adaptive attack to evaluate the robustness of models. We show the experiment results of Cora-ML and Citeseer, averaged over 5 different random splits, in Table 7 and Table 30 (in Appendix Q), respectively. The ablation studies on the layers and γ in MCP are presented in Table 31. Please refer to Appendix Q for more detailed results and studies. From the results, we can conclude that our Pro-GAT significantly outperforms the backbone GAT and exhibits strong robustness across various budgets while keeping good clean accuracy.

Table 7: Semi-supervised node classification on Cora-ML.

Model \ Budget	0%	10%	20%	30%	40%
GCN	85.0 ± 0.4	69.6 ± 0.5	60.9 ± 0.7	54.2 ± 0.6	48.4 ± 0.5
GNNGuard	83.1 ± 0.7	70.2 ± 1.0	63.1 ± 1.1	57.5 ± 1.6	51.0 ± 1.2
RGCN	<u>85.7 ± 0.4</u>	69.1 ± 0.4	59.8 ± 0.7	52.8 ± 0.7	46.1 ± 0.7
GRAND	**86.1 ± 0.7**	70.7 ± 0.7	61.6 ± 0.7	56.7 ± 0.8	51.9 ± 0.9
ProGNN	85.6 ± 0.5	71.0 ± 0.5	63.0 ± 0.7	56.8 ± 0.7	51.3 ± 0.6
Jaccard-GCN	83.7 ± 0.7	68.3 ± 0.7	60.0 ± 1.1	54.0 ± 1.7	49.1 ± 2.4
SoftMedian	85.0 ± 0.7	**75.5 ± 0.9**	<u>69.5 ± 0.5</u>	<u>62.8 ± 0.8</u>	<u>58.1 ± 0.7</u>
GAT	83.5 ± 0.5	<u>71.2 ± 1.2</u>	65.0 ± 0.9	60.5 ± 0.9	56.7 ± 0.9
Pro-GAT (ours)	84.6 ± 0.8	**75.5 ± 0.8**	**72.1 ± 0.4**	**69.0 ± 0.7**	**66.5 ± 1.2**

F THE DERIVATION AND CONVERGENCE GUARANTEE OF NEWTON-IRLS ALGORITHM

Despite of several robustness implications introduced by the penalties, the proposed robust estimator in Eq. (5) is non-convex and non-smooth, posing a significant challenge for designing an efficient algorithm to solve it. There have been several optimization algorithms designed for this type of problems, such as the Alternating Direction Multiplier Method (ADMM) Glowinski & Marroco (1975); Gabay & Mercier (1976), Iteratively Reweighted Least Squares (IRLS) algorithm Daubechies et al. (2008), Newton-type approaches (Fan & Li, 2001; Zhang, 2010), and coordinate descent algorithms Breheny & Huang (2011). However, these algorithms typically require excessive computation and memory space, and they are incompatible with back-propagation in training deep learning models. Therefore, it is infeasible to directly incorporate these algorithms into the transformer architectures.

To this end, we propose an efficient Newton iterative reweighted least square (Newton-IRLS) algorithm to tackle this intractable problem. We first design a localized upper bound for the original objective and then optimize the upper bound with a second-order Newton method with rigorous theoretical loss descent guarantee. The precise statements are presented as follows and the detailed proof are provided in Appendix G.

Localized upper bound. Instead of directly optimizing the original loss function $\mathcal{L}(\boldsymbol{z})$, we optimize a convex localized upper bound at the current iteration $\boldsymbol{z}^{(k)}$ as follows:

Lemma F.1 (Localized Upper Bound)**.** *Suppose the loss objective is defined as in Eq. (5), where $\rho \circ sqrt(\cdot)$ is any non-convex function. For any fixed point $\boldsymbol{z}^{(k)}$, there exists a convex localized upper bound as:*

$$\hat{\mathcal{L}}(\boldsymbol{z}) = \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2 + C(\boldsymbol{z}^{(k)}), \tag{7}$$

where $w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}$ and ρ' is the first derivative of ρ. Particularly, the constant $C(\boldsymbol{z}^{(k)})$ guarantees the equality of $\hat{\mathcal{L}}$ and $\mathcal{L}$ at $\boldsymbol{z}^{(k)}$, i.e., $\hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \mathcal{L}(\boldsymbol{z}^{(k)})$.

Proof. Please refer to Appendix G.1. □

As $C(\boldsymbol{z}^{(k)})$ is treated as a constant during the optimization at the current step, the upper bound in Eq. (7) becomes convex and can be efficiently optimized.

Newton-IRLS iteration. After obtaining the convex upper bound in Eq. (7), it is easy to employ the convex optimization algorithm to solve it. Typically, we use the first-order gradient descent or the second-order Newton method to optimize the objective. In most of the cases, second-order methods converge faster than the first-order ones, but require substantial computations due to the calculations of Hessian matrix, matrix inversion and multiplication. However, due to the uniqueness of our $\hat{\mathcal{L}}$ in Eq. (7), we can derive a concise closed-form iteration using the second-order Newton method as follows:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} - \left[\nabla^2 \hat{\mathcal{L}}(\boldsymbol{z}^{(k)})\right]^{-1} \nabla \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \frac{\sum_j a_j \cdot w_j^{(k)} \cdot \boldsymbol{v}_j}{\sum_j a_j \cdot w_j^{(k)}}. \tag{8}$$

Proof. Please refer to Appendix G.3. □

Theorem F.2 (Convergence guarantee)**.** *Suppose the loss objective $\mathcal{L}(\boldsymbol{z})$ is defined as in Eq. (5) and its corresponding convex localized upper bound is in Eq. 7. Then, through the iteration in Eq. (8), the following inequality holds:*

$$\mathcal{L}(\boldsymbol{z}^{(k+1)}) \leq \hat{\mathcal{L}}(\boldsymbol{z}^{(k+1)}) \leq \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \mathcal{L}(\boldsymbol{z}^{(k)}),$$

that is, optimizing upper bound $\hat{\mathcal{L}}$ can guarantee the rigorous descent of $\mathcal{L}$.

Proof. Please refer to Appendix G.2. □

G PROOF OF NEWTON-IRLS ALGORITHM

G.1 PROOF OF LOCALIZED UPPER BOUND (LEMMA F.1)

Proof. Define $\phi(z) := \rho(\sqrt{z})$ as a non-convex function, then for fixed point z_0,

$$\phi(z) \leq \phi(z_0) + \phi'(z_0)(z - z_0) = \phi'(z_0) \cdot z + C(z_0)$$

where the first inequality holds with equality at $z = z_0$ and

$$\phi'(z_0) = \rho'(\sqrt{z}) \cdot \frac{1}{2\sqrt{z}}\bigg|_{z=z_0} = \frac{\rho'(\sqrt{z_0})}{2\sqrt{z_0}}.$$

By replacemnet as $z = \|\boldsymbol{v}_j - \boldsymbol{z}\|^2$ and $z_0 = \|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|^2$, then

$$\begin{aligned}\rho(\|\boldsymbol{v}_j - \boldsymbol{z}\|) &\leq \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2 + C(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|^2) \\ &= w_j^{(k)} \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2 + C(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|^2),\end{aligned}$$

and the first inequality holds with equality at $\boldsymbol{z} = \boldsymbol{z}^{(k)}$. Sum up the items on both sides with weights $\{a_j\}_{j\in[N]}$, we obtain

$$\begin{aligned}\mathcal{L}(\boldsymbol{z}) &= \sum_{j=1}^{N} a_i \cdot \rho(\|\boldsymbol{v}_j - \boldsymbol{z}\|) \\ &\leq \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2 + \sum_{j=1}^{N} a_j \cdot C(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|^2) \\ &= \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot \|\boldsymbol{v}_j - \boldsymbol{z}\|^2 + C_1(\boldsymbol{z}^{(k)}) \\ &= \hat{\mathcal{L}}(\boldsymbol{z})\end{aligned} \tag{9}$$

and the equality holds at $\boldsymbol{z} = \boldsymbol{z}^{(k)}$:

$$\hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \mathcal{L}(\boldsymbol{z}^{(k)}). \tag{10}$$

□

After obtaining the convex upper bound $\hat{\mathcal{L}}(\boldsymbol{z})$, it becomes feasible to employ convex optimization algorithms to optimize this objective.

G.2 PROOF OF RIGOROUS LOSS DESCENT GUARANTEE (THEOREM F.2)

Proof. Since $\boldsymbol{z}^{(k+1)}$ is obtained from optimize the convex localized upper bound $\mathcal{L}$ at $\boldsymbol{z}^{(k)}$, then we have $\hat{\mathcal{L}}(\boldsymbol{z}^{(k+1)}) \leq \hat{\mathcal{L}}(\boldsymbol{z}^{(k)})$. According to the upper bound in Eq. (9) and localized equality in Eq. (10), it is not hard to get the following inequality:

$$\mathcal{L}(\boldsymbol{z}^{(k+1)}) \leq \hat{\mathcal{L}}(\boldsymbol{z}^{(k+1)}) \leq \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \mathcal{L}(\boldsymbol{z}^{(k)}).$$

Therefore, optimizing the localized upper bound $\hat{\mathcal{L}}$ can guarantee the rigorous descent of $\mathcal{L}$.

□

G.3 PROOF OF NEWTON-IRLS ALGORITHM AND SPECIAL CASES

Newton-IRLS. We first derive the formulations of gradient and Hessain matirx of $\hat{\mathcal{L}}$ as follows:

$$\nabla\hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot 2(\boldsymbol{z}^{(k)} - \boldsymbol{v}_i)$$

$$\nabla^2 \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) = \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot 2 \cdot \mathbf{I}$$

Then, the gradient descent (GD) (η is the stepsize) is

$$\begin{aligned} \boldsymbol{z}^{(k+1)} &= \boldsymbol{z}^{(k)} - \eta \cdot \nabla \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) \\ &= \boldsymbol{z}^{(k)} - \eta \cdot \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot 2(\boldsymbol{z}^{(k)} - \boldsymbol{v}_j), \end{aligned}$$

and the Newton Iteration is

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} - \left[\nabla^2 \hat{\mathcal{L}}(\boldsymbol{z}^{(k)})\right]^{-1} \nabla \hat{\mathcal{L}}(\boldsymbol{z}^{(k)}) \tag{11}$$

$$= \boldsymbol{z}^{(k)} - \cdot \left(\sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot 2 \cdot \mathbf{I} \right)^{-1} \sum_{j=1}^{N} a_j \cdot w_j^{(k)} \cdot 2(\boldsymbol{z}^{(k)} - \boldsymbol{v}_i) \tag{12}$$

$$= \frac{\sum_j a_j \cdot w_j^{(k)} \cdot \boldsymbol{v}_j}{\sum_j a_j \cdot w_j^{(k)}} \tag{13}$$

In convex optimization, it has been well-established that second-order methods converge much faster than first-order approaches, but they require substantial computation in calculating or approximating the inverse Hessian matrix. However, due to the uniqueness of our $\hat{\mathcal{L}}$ in Eq. (7), we can derive a concise closed-form iteration using the second-order Newton method as in Eq. (13). Compared to the first-order gradient descent (GD) iteration, our Newton-IRLS algorithm enjoys several advantages as follows:

- Fast convergence: Newton method converges at a quadratic rate, which is significantly faster than the linear convergence of gradient descent (GD). The comparative analysis of them can be found in Figure 6 (a) in ablation studies;
- Interpretable formulation: The resulted form in Eq. (8) employs a normalized reweighted sum, which can be interpreted as robust estimator by down-weighting the outliers, as discussed in the following paragraph;
- Efficient computation: The Hessian $\nabla^2 \hat{\mathcal{L}}(\boldsymbol{z}^{(k)})$ can be easily computed as a closed-form diagonal matrix, facilitating the matrix inversion and multiplication in the Newton's iteration.

H Special cases of Newton-IRLS

Our Newton-IRLS is a general framework which can be derived as different reweighting schemes with different penalties:

- Square Loss (ℓ_2):

$$\rho(z) = \frac{1}{2} z^2,$$

$$w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} = \frac{1}{2},$$

$$\boldsymbol{z}^* = \sum_{j=1}^{N} a_j \cdot \boldsymbol{v}_j.$$

- Absolute Loss (ℓ_1):

$$\rho(z) = z,$$

$$w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} = \frac{1}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}.$$

- Minimax Concave Penalty (MCP) Zhang (2010):

$$\rho_\gamma(z) = \begin{cases} z - \frac{z^2}{2\gamma} & \text{if } y < \gamma \\ \frac{\gamma}{2} & \text{if } y \geq \gamma \end{cases},$$

$$w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} = \frac{1}{2} \max\left[\frac{1}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} - \frac{1}{\gamma}, 0\right].$$

- Huber loss:

$$\rho_\delta(z) = \begin{cases} \frac{1}{2}z^2 & \text{if } z < \delta \\ \delta \cdot (z - \frac{1}{2}\delta) & \text{if } z \geq \delta \end{cases},$$

$$w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} = \frac{1}{2} \min\left[1, \frac{\delta}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|}\right].$$

- Huber-MCP:

$$\rho_{\delta,\gamma}(z) = \begin{cases} \frac{1}{2}z^2 & \text{if } z < \delta \\ \delta \cdot (z - \frac{1}{2}\delta - \frac{(z-\delta)^2}{2(\gamma-\delta)}) & \text{if } \delta \leq z < \gamma \\ \frac{\delta\gamma}{2} & \text{if } \gamma \leq z \end{cases},$$

$$w_j^{(k)} = \frac{\rho'(\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|)}{2\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} = \frac{1}{2} \max\left[\min\left[\frac{\delta}{\gamma - \delta}\left(\frac{\gamma}{\|\boldsymbol{v}_j - \boldsymbol{z}^{(k)}\|} - 1\right), 1\right], 0\right].$$

I Dataset Information

I.1 Language Domain

- **AG's News Corpus (AGNEWS)** Zhang et al. (2015): It is a collection of more than 1 million news articles. News articles have been gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. ComeToMyHead is an academic news search engine which has been running since July, 2004. The dataset is provided by the academic comunity for research purposes in data mining (clustering, classification, etc), information retrieval (ranking, search, etc), xml, data compression, data streaming, and any other non-commercial 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.
- **Internet Movie Database (IMDB)** Maas et al. (2011): IMDB dataset having 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing. So, predict the number of positive and negative reviews using either classification or deep learning algorithms.
- **Stanford Sentiment Treebank (SST-2)** Socher et al. (2013): It is a corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. Binary classification experiments on full sentences (negative or somewhat negative vs somewhat positive or positive with neutral sentences discarded) refer to the dataset as SST-2 or SST binary.
- **Recognizing Textual Entailment (RTE)**: It comes from a series of annual textual entailment challenges. The authors of the benchmark combined the data from RTE1 Dagan et al. (2010), RTE2 Bar-Haim et al. (2014), RTE3 Giampiccolo et al. (2007), and RTE5 Bentivogli et al. (2009). Examples are constructed based on news and Wikipedia text. The authors of the benchmark convert all datasets to a two-class split, where for three-class datasets they collapse neutral and contradiction into not entailment, for consistency.
- **Behaviors**: It is a new dataset introduced in Zou et al. (2023) for robustness evaluation of jailbreaking attack. The dataset includes 520 goal prompts and corresponding targets, it is available in https://github.com/llm-attacks/llm-attacks/blob/main/data/advbench/.

I.2 Beyond Language Domain

- **CIFAR10** Krizhevsky et al. (2009): The CIFAR-10 dataset is a well-known dataset used in the field of computer vision. It consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is divided into two parts: 50,000 training images and 10,000 test images. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each image is labeled with one of these 10 categories.
- **Cora-ML** Sen et al. (2008): The Cora dataset is a widely-used benchmark dataset in the field of graph-based tasks. It consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words. Working with the Cora dataset presents challenges typical of real-world graph data, such as handling sparse and high-dimensional feature vectors, and dealing with the complex structure of the graph.
- **Citeseer** Giles et al. (1998): The CiteSeer dataset is another popular dataset in the graph field. It consists of 3312 scientific publications classified into one of six classes. The citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words.

J Defense Baselines and Backbone Architectures

J.1 Defense Baselines

Language Domain:

- **PGD-Adv** Madry et al. (2018): The Projected Gradient Descent (PGD) method stands as the most prevalent attack strategy in the field of computer vision. It is primarily utilized for crafting adversarial examples in the context of adversarial training. The defense in this paper is adapted directly from PGD-adv in computer vision, extending its application to language modeling.
- **MixADA** Si et al. (2021): The search space for adversarial examples in language models is typically vast due to their discrete nature. To enhance the robustness of these models, MixADA integrates adversarial training Goodfellow et al. (2014) with mixup data augmentation Zhang et al. (2018), thereby expanding the range of adversarial examples covered. Specifically, mixup generates synthetic training examples by linearly blending pairs of inputs and their corresponding labels. This approach enables the model to learn from a broader and more effective set of adversarial examples during training.
- **FreeLB** Zhu et al. (2020): Different from attacks that directly change the words in the sentence, FreeLB adds adversarial perturbations to word embeddings and minimizes the resultant adversarial loss around input samples. To expedite the process of adversarial training, FreeLB implements a single descent step on the parameters concurrently with each of the K ascent steps applied to the perturbation, which utilizes the average of accumulated gradients over the K steps. This efficiency has established FreeLB as a popular defense method in the field of NLP.
- **TA-VAT** Li & Qiu (2021): TA-VAT is another virtual adversarial training method that generates gradient-based perturbations on the embedding space. To create fine-grained perturbations, TA-VAT employs a token-level accumulated perturbation vocabulary. This vocabulary serves to better initialize the perturbations. Additionally, TA-VAT utilizes a token-level normalization ball, which effectively constrains these perturbations in a relevant and precise manner.
- **Adversarial Training (AT)**: Adversarial training is adaptive to the attack to be evaluated. Take the Textfooler as the instance, at every epoch, we generate 1000 perturbations from the Textfooler and add them into the training dataset to reinforce the training of models. We utilize the TextAttack Morris et al. (2020) platform the conduct this adversarial training.
- **SmoothLLM** Robey et al. (2023): Motivated by finding that the adversarial-prompting jailbreak is sensitive to the random character-level changes, SmoothLLM is designed by firstly perturbing multiple copies of the given prompt and then aggregating all the outputs.

Beyond Language Domain:

- **Graph Convolutional Network (GCN)** Kipf & Welling (2017): GCN is motivated by the localized first-order approximation of spectral graph convolutions. The basic idea is to first add self-loops to the adjacency matrix and then normalize the matrix.
- **Graph Attention Network (GAT)** Veličković et al. (2018): GAT leverages the attention mechanism to construct masked self-attentional layers. This allows the nodes to reweight their neighbors via the feature similarity.
- **GNNGuard** Zhang & Zitnik (2020): GNNGuard is a universal reweighting framework that can be applied to any GNN. It leverages the cosine similarities between nodes' features to up-weight the correlated nodes and prune the edges between the dissimilar pairs.
- **Robust GCN (RGCN)** Zhu et al. (2019): RGCN first models the latent representations as the Gaussian distributions. Then the weights of different neighborhoods will be assigned different weights according to their variances when performing the message propagation.
- **Graph Random Neural Network (GRAND)** Feng et al. (2020): The core of GRAND is the random propagation, wherein the node feature will be partially or entirely dropped out and then propagated through over the graph. This operation enable the node to be insensitive to

the specific neighborhood, which prevents the effect of malicious outliers. Additionally, the random propagation also help to augment the representation for each node, thus improving the generalization of GNN.

- **Property GNN (ProGNN)** Jin et al. (2020b): The core principle of ProGNN is to robustify the GNNs through enhancing the graph properties of sparsity, low rank and feature smoothness. It provides a graph structure learning framework to learn the clean graph structure and parameters simultaneously.
- **Jaccard-GCN** Wu et al. (2019): The basic idea of Jaccard-GCN is to preprocess the adjacency matrix by first computing the Jaccard coefficients of paired node features and then dropping the edges where the coefficients are below the threshold.
- **SoftMedian** Geisler et al. (2021): SoftMedian is a robust estimator for the message passing aggregation. It reweights the adjacency weights based to the distances of the hidden embeddings between the neighbor nodes and the dimension-wise median of the the entire neighboring representations.

J.2 Backbone Architectures

Classical language models:

- **BERT** Devlin et al. (2018): BERT stands out as one of the most well-known transformer-based language models. It is pretrained through masked language modeling (MLM), where it learns to predict words that have been masked, using context for guidance. This pretrained model is then fine-tuned for a variety of downstream tasks, showcasing its versatility and effectiveness in diverse applications. In our experiments, we will use BERT-110M.
- **RoBERTa** Liu et al. (2019): RoBERTa is developed to overcome certain limitations of the original BERT model. This is accomplished by implementing key modifications such as increasing the batch size, extending the training epochs, and employing advanced optimization techniques. As a result of these strategic changes, RoBERTa has demonstrated substantial performance improvements over BERT across various NLP benchmarks. In our experiments, we will use RoBERTa-125M.
- **ALBERT** Lan et al. (2020): ALBERT is a lite variant of BERT. It is achieved by decoupling the word embedding from the hidden embedding, significantly cutting down the number of parameters. To further enhance its efficiency, ALBERT employs cross-layer parameter sharing, ensuring that all layers use the same parameters. The reductions not only minimize memory footprint but also improve the efficiency of the model. In our experiments, we will use ALBERT-12M.
- **DistilBERT** Sanh et al. (2019): DistilBERT is a light version of BERT, maintaining most of the performance of the original BERT. It is trained with the knowledge distillation technique Hinton et al. (2015) to achieve high efficiency. In our experiments, we will use DistilBERT-66M.

Large Language Models:

- **T5** Raffel et al. (2023): Text-to-Text Transfer Transformer (T5) is a transformer-based neural network model known for its versatility and power in handling a wide range of NLP tasks. T5 simplifies NLP tasks by treating them uniformly as text-generation challenges. The T5 model family offers a range of sizes, from 60 million to 11 billion parameters, catering to different computational needs. The flexibility has made T5 a popular choice in NLP research. In our experiments, we will use T5-770M.
- **LLaMA** Touvron et al. (2023): LLaMa, the Large Language Model developed by Meta AI, represents a cutting-edge advancement in language modeling. Trained on publicly available datasets, LLaMa is available in various sizes to suit different computational needs. Notably, LLaMa-13B demonstrates superior performance over GPT-3 in most benchmarks, highlighting its exceptional effectiveness and capability in NLP tasks. In our experiments, we will use LLaMA-7B.
- **Vicuna** Chiang et al. (2023): Vicuna is a high-performing, open-source chatbot that impresses with capabilities comparable to GPT-4. Fine-tuned from the LLaMa model, it utilizes

user-shared conversations gathered from Share-GPT for its training. Remarkably, Vicuna achieves 90% of the performance level of GPT-4, despite having only 13 billion parameters, showcasing its efficiency and effectiveness. In our experiments, we will use Vicuna-7B.

Vision Models:

- **ViT** Dosovitskiy et al. (2020): The Vision Transformer (ViT) is a model in computer vision that adopts the principles of the Transformer architecture. In ViT, an image is processed similarly to a sequence of words, or tokens. Specifically, the image is segmented into fixed-size patches, each of which is then linearly transformed into an embedded representation. When trained on sufficient data, ViT achieves state-of-the-art performance on image classification benchmarks, competing with or outperforming leading CNN-based models. In our experiments, we will use ViT-86M.
- **Swin** Liu et al. (2021b): Swin Transformer is a popular variant of ViT, standing out for its enhanced efficiency and superior performance. It employs a hierarchical architecture, which not only aligns more closely with the nature of visual data but also boosts efficiency. To effectively capture global contextual information, Swin Transformer incorporates shifted window-based self-attention, further enhancing its effectiveness in vision-related applications. In our experiments, we will use Swin-50M.
- **BEIT** Bao et al. (2022): Due to the success of BERT, BEIT harnesses the concept of masked language modeling to enhance self-supervised learning in the visual domain. To align with the words in language models, BEIT first maps the patch in an image into a token with an autoencoder. In the training process, it masks a portion of these patches, using the remaining unmasked ones to predict the masked tokens. Subsequently, the model is fine-tuned for a variety of downstream tasks, demonstrating its adaptability and effectiveness in diverse applications. In our experiments, we will use BEIT-86M.
- **DeiT** Touvron et al. (2021): To address the substantial data requirements for training the Vision Transformer, Data-Efficient Image Transformer (DeiT) employs knowledge distillation Hinton et al. (2015) to train the model. By integrating this approach with various data augmentation techniques, DeiT successfully attains competitive results in image classification tasks, even with constrained training data availability. In our experiments, we will use DeiT-22M.
- **ConViT** d'Ascoli et al. (2021): ConViT designs a hybrid architecture to leverage the local processing capabilities of CNNs and the global context understanding of transformers. To be specific, it replaces the several first self-attention layers with gated-self positional self-attention layers, allowing the model to adjust between local and global processing. In our experiments, we will use ConViT-30M.

K ATTACKS.

K.1 CLASSIC TEXT ATTACK:

For the classic text attacks, we follow the default attack setting in the TextAttack Morris et al. (2020) and the detailed information are as follows:

- **DeepWordBug** Gao et al. (2018): DeepWordBug is black-box attacks that apply character-level transformations to the highest-ranked tokens misclassify the text input. It includes several character transformations including swap, substitution, deletion and insertion. We hold the maximum difference on edit distance (Levenshtein Edit Distance) to 30 for each sample. We will greedily modify the works with the word importance ranking.
- **PWWS** Ren et al. (2019): The probability weighted word saliency (PWWS) employs a new word order determined by the word saliency and predicted probability, and then greedily perform the synonyms substitution.
- **TextFooler** Jin et al. (2020a): TextFooler propose a more comprehensive paradigm to generate adversarial perturbations. It firstly identify the important words and then replace them with the most semantically and syntacticaly similar synonyms until the prediction is altered. We set the minimum word embedding cosine similarity as 0.5 and the universal sentence similarity threshold as 0.840845057.
- **TextBugger** Li et al. (2018): TextBugger is a framework for creating utility-preserving adversarial texts against the classification models. It can be employed under both white-box and black-box settings. Specifically, this attack sorts the words based on Jacobian matrix in white-box setting and based on specific scoring function in black-box setting. It includes swap, substitution, deletion and insertion. There are two types of substitutions: Sub-C (replace characters with visually similar characters) and Sub-W (replace a word with its top-K ($K = 5$) nearest neighbors.) The universal sentence threshold is set as 0.8.

K.2 ATTACK IN LLMS

- **Prompt attack**: we conduct adversarial prompt attacks using the text attacks including DeepWordBug Gao et al. (2018), PWWS Ren et al. (2019), TextFooler Jin et al. (2020a) and TextBugger Li et al. (2018). The evaluation setting follow PromptBench Zhu et al. (2024) which is a unified and comprehensive python library for evaluating LLMs.
- **Jailbreak**: We employ the Greedy Coordinate Gradient (GCG) Zou et al. (2023) method to search for the adversarial prompt suffixes that circumvent the alignment of the language models. It can be conducted under the transfer attack and direct attack. Specifically, the transfer attacks firstly attack the surrogate model (e.g., Vicuna) and can apply the perturbations to the victim models. The direct attacks directly apply the GCG method to the victim models to generate the suffixes, which can be much stronger. The JailbreakKeywords is a list of words commonly found in messages that abstain from responding to a prompt asking for inappropriate content. In our experiment, we use the following list: JailbreakKeywords = ["I'm sorry", "Sorry", "I apologize", "As an", "I'm just", "I'm an", "I cannot", "I would advise", "it's not appropriate", "As a responsible AI"].

K.3 ATTACKS IN VISION & GRAPH MODELS

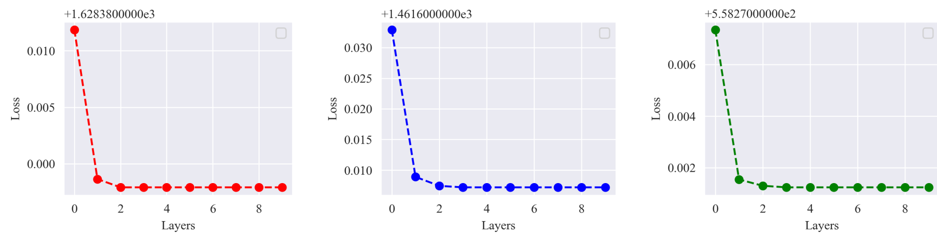
- **FGSM** Goodfellow et al. (2014): Fast Gradient Sign Method (FGSM) is a widely used technique for generating adversarial examples. It crafts perturbations specifically to maximize errors in the output of neural networks. This is achieved by analyzing the sign (positive or negative) of each element in the gradient. Due to its one-step process, FGSM is highly efficient in generating these adversarial examples.
- **PGD** Madry et al. (2018): Projected Gradient Descent (PGD) method is one of the most prominent attack strategies in computer vision. Unlike FGSM employing a single step, PGD uses multiple steps to generate adversarial examples. This iterative approach includes a projection operation, which ensures that the intensity of the attack remains within specified limits, making PGD a more controlled and effective method for generating adversarial examples. The steps are $K = 7$ and the steps size $\alpha = 0.00784$.

- **PGD on Graph** Xu et al. (2019): Motivated by PGD Madry et al. (2018) in vision domain, Xu et al. (2019) propose a first-order method to conduct topology attack on discrete graph structure. This method firstly solve continuous optimization problem by Projected Gradient Descent (PGD) method and then utilize the random sampling to get the optimal binary topology perturbation from the continuous probabilistic matrix.

L Algorithm Convergence and Robust Estimation

L.1 Convergence Guarantee

Loss curves. We use generated data to verify the convergence of our proposed algorithm. The batch size, number of heads, length of inputs and dimension of data are chosen as $B = 8, H = 4, N = 64, D = 8$, respectively. The γ in MCP is set as 4 and δ in Huber loss is set as 0.8. The loss curve of our algorithm with different penalties are shown in Figure 9. We can observe that our algorithm show a fast convergence and even 2 to 3 steps can well approximate the optimal solution.



(a) ℓ_1 (b) MCP (c) Huber

Figure 9: Loss Curve of Algorithms

Trajectory. To further validate the convergence and effectiveness of our algorithm, we use a toy experiment to visualize the trajectories of updated vector in 2D plane in Figure 10. We use L_1 penalty in our algorithm, the simulated attention matrix and value matrix are as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \mathbf{V} = \begin{bmatrix} 1 & 2 \\ 7 & 25 \\ 25 & 37 \end{bmatrix}. \tag{14}$$

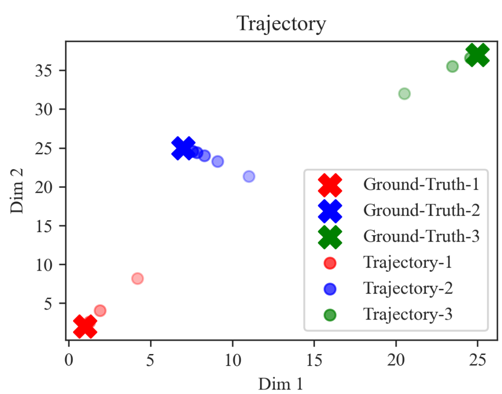


Figure 10: Optimization trajectory.

From the figure, we can find that with the mean as the initial position, the updated vector can approach closely to the ground truth within only 3 steps. This phenomenon further validate the effectiveness and efficiency of our algorithm.

L.2 Robust Estimation

Robust estimation. We firstly generated clean samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$ (blue dots) and the outlier samples $\{\boldsymbol{x}_i\}_{i=n+1}^{n+m}$(red dots) from 2-dimensional Gaussian distributions, $\mathcal{N}((0,0),1)$ and $\mathcal{N}((8,8),0.5)$, respectively. We calculate the mean of clean samples $\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$ as the ground truth of the mean estimator. Then we estimate the mean of all the samples by solving $\arg\min_{\boldsymbol{z}} \sum_{i=1}^{n+m} \rho(\boldsymbol{z} - \boldsymbol{x}_i)$ using the our method, where $\rho(\cdot)$ can take different penalties such as ℓ_2 penalty $\|\cdot\|_2^2$ and ℓ_1 penalty $\|\cdot\|_2$.

In Figure 5, we visualize the generated clean samples and outliers, as well as the ground truth means and the mean estimators with $\eta(\cdot) = \|\cdot\|_2^2$ or $\|\cdot\|_2$ under different outlier ratios 15% and 45%. The results show that, with the existence of outliers, the ℓ_2-based estimator deviates far from the true mean, while the ℓ_1-based estimator is more resistant to outliers and MCP-based estimator is the most robust.

M ADDITIONAL EXPERIMENTS OF TEXT ATTACKS

M.1 SENTIMENT ANALYSIS: IMDB

We present the results of sentiment analysis on IMDB dataset under various attacks in Table 8. We can conlude from the results that our methods improve the robustness of the backbones significantly by simply plugging the ProAttention into the models without additional fintuning or training. Moreover, our method can be combined with the existing defenses such as Adversarial Training (AT) to further improve the performance.

Table 8: The results of sentiment analysis on IMDB.

		TEXTFOOLER		TEXTBUGGER		DEEPWORDBUG		PWWS	
MODEL	CLEAN% ↑	AUA% ↑	ASR% ↓	AUA% ↑	ASR% ↓	AUA% ↑	ASR% ↓	AUA% ↑	ASR% ↓
ROBERTA	93.3	23.7	74.6	9.4	89.9	36.5	60.9	19.5	79.1
DISTILBERT	90.9	14.9	83.6	4.3	95.3	18.8	79.3	9.6	89.4
ALBERT	92.8	21.8	76.5	14.1	84.8	36.2	61.0	15.9	82.9
BERT	92.3	11.8	87.2	11.3	87.4	32.8	64.5	26.4	71.5
FREELB	93.0	25.1	73.6	19.9	76.9	40.9	55.5	42.0	54.7
PGD	93.2	26.2	69.2	17.4	81.6	32.0	65.8	27.2	69.6
MIXADA	91.9	16.7	82.0	11.8	87.3	33.4	65.8	30.0	67.4
TA-VAT	93.0	28.5	67.6	27.3	68.8	34.7	60.4	35.1	59.8
AT	93.2	33.6	64.3	31.8	66.1	37.7	61.5	28.7	70.3
PRO-BERT (ℓ_1) (OURS)	93.3	24.6	73.6	13.0	86.1	36.0	61.4	32.7	65.0
PRO-BERT (HUBER) (OURS)	93.0	24.8	73.3	13.4	85.6	36.9	60.3	31.5	66.1
PRO-BERT (MCP) (OURS)	93.5	22.1	76.9	44.6	53.2	55.5	41.8	56.3	41.1
PRO-BERT (MCP) + AT (OURS)	**93.6**	**42.0**	**56.1**	**55.3**	**41.0**	**60.8**	**39.0**	**61.0**	**37.6**

M.2 TEXTUAL ENTAILMENT: RTE

In Table 9, we display the results of textual entailment on RTE across different cosine similarities constraints in TextFooler attack. We select DistilBERT as the backbone model and construct several MCP-based architectures with different γ. We can observe that our method can improve the robustness acorss different cosine similarities. The performance improvement is more evident under the smaller cosine similarities, which is equivalent to larger budgets.

In Table 10, we present the results of textual entailment on RTE across various attacks. The results exhibit the consistent improvement of our methods over the backbone model.

Table 9: The results of textual entailment on RTE across different cosine similarities in TextFooler .

	COS-SIM	0.5		0.6		0.7		0.8	
MODEL	CLEAN%	AUA%	ASR%	AUA%	ASR%	AUA%	ASR%	AUA%	ASR%
DISTILBERT	62.5	4.0	93.6	5.1	91.9	7.9	87.3	18.1	71.1
PRO-DISTILBERT-MCP $\gamma = 0.2$	63.3	6.9	89.5	6.1	90.6	9.4	85.6	18.1	72.4
PRO-DISTILBERT-MCP $\gamma = 0.3$	62.4	15.2	75.3	16.6	72.9	20.6	66.5	30.7	50.0
PRO-DISTILBERT-MCP $\gamma = 0.4$	56.0	18.1	67.7	24.6	56.1	28.2	49.7	33.2	40.7
PRO-DISTILBERT-MCP $\gamma = 0.5$	55.6	15.9	70.1	17.7	66.7	20.2	61.9	28.9	45.6
PRO-DISTILBERT-MCP $\gamma = 0.6$	53.1	10.8	80.5	12.3	77.9	16.3	70.8	20.2	63.6

Table 10: The results of textual entailment on RTE across different attacks.

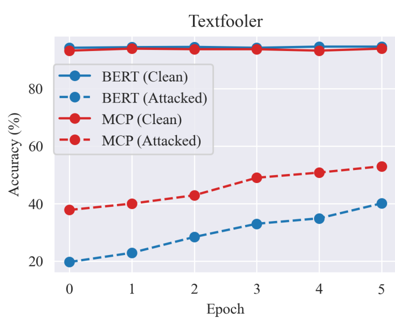
	ATTACK	TEXTFOOLER		TEXTBUGGER		DEEPWORDBUG		PWWS	
MODEL	CLEAN%	AUA%	ASR%	AUA%	ASR%	AUA%	ASR%	AUA%	ASR%
DISTILBERT	62.5	7.9	87.3	3.6	94.2	18.4	70.5	12.3	80.4
PRO-DISTILLBERT-MCP	63.3	28.2	49.7	14.4	74.5	33.6	40.0	24.2	60.6

M.3 Adversarial Fine-tuning on Topic Classification: AGNEWS

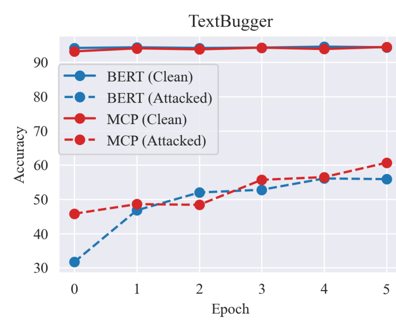
Adversarial training techniques are highly effective to enhance the robustness of models via adding the adversarial examples into the training set. To better capture the robustness enhancement of adversarial training, we track the adversarial fine-tuning curves and present the detailed results on AG's News in Table 11 and Figure 11. In the beginning of every epoch, we generate 1000 perturbed examples using the specific attack and then put them to the original training dataset. From the results, we can make the following observations: (1) the models show even better robustness during the process of adversarial training. (2) our method can be combined with adversarial training to further improve the resilience of the models.

Table 11: Adversarial fine-tuning on AGNEWS.

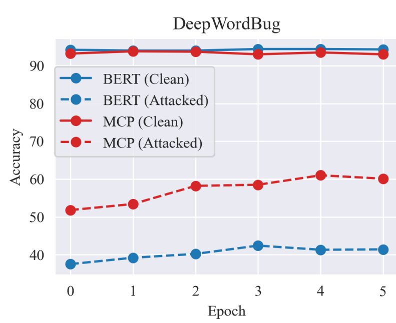
Model	Clean%	AUA%(TextFooler)	Clean%	AUA%(TextBugger)	Clean%	AUA%(DeepWordBug)	Clean%	AUA%(PWWS)
BERT epoch-0	94.2	19.7	94.2	31.7	94.2	37.5	94.2	43.1
BERT epoch-1	94.4	22.9	94.4	46.8	94.0	39.2	94.5	49.3
BERT epoch-2	94.5	28.4	94.2	52.0	94.0	40.2	94.4	54.4
BERT epoch-3	94.2	33.0	94.3	52.8	94.4	42.4	94.1	57.3
BERT epoch-4	94.6	34.9	94.6	56.1	94.4	41.3	93.8	62.6
BERT epoch-5	94.6	40.1	94.4	55.9	94.3	41.4	93.8	59.3
Pro-BERT-MCP epoch-0	93.2	37.8	93.2	45.8	93.2	51.8	92.2	55.0
Pro-BERT-MCP epoch-1	93.9	40.0	94.1	48.6	93.8	53.4	93.4	57.5
Pro-BERT-MCP epoch-2	93.7	42.9	93.8	48.4	93.7	58.2	93.6	59.9
Pro-BERT-MCP epoch-3	93.7	49.0	94.3	55.7	93.0	58.5	93.0	65.2
Pro-BERT-MCP epoch-4	93.2	50.8	93.9	56.5	93.5	61.0	93.6	65.1
Pro-BERT-MCP epoch-5	93.9	53.0	94.5	60.7	93.0	60.1	93.6	68.8



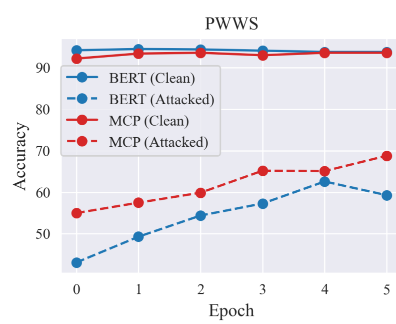
(a) TextFooler



(b) TextBugger



(c) DeepWordBug



(d) PWWS

Figure 11: Adversarial fine-tuning on AGNEWS.

M.4 Ablation Study on Attack constraints

We present the ablation study on the maximum perturbation percentage, minimum cosines similarity and sentence similarity threshold in Figure 12, Table 12, Table 13 and Table 14, respectively. The experiments are performed on AGNEWS under TextFooler with the ALBERT as the backbone. The default values are as follows: sentence similarity threshold is 0.840845057, maximum perturbation percentage is 1.0, synonym cosine similarity is 0.5. The results show the consistent improvement of our method over the backbone models.

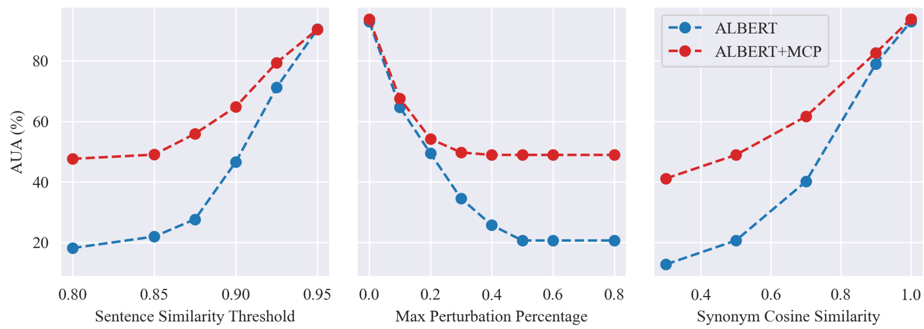


Figure 12: Ablation studies on attack constraints.

Table 12: Ablation study on max perturbation percentage.

Max-Percentage (ρ)		0.1		0.2		0.3		0.4	
Model	Clean%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%
ALBERT	93.0	64.7	30.4	49.4	46.9	34.5	62.9	25.7	72.4
Pro-ALBERT-MCP	93.8	67.6	27.2	54.2	41.6	49.7	46.4	48.9	47.3

Max-Percentage (ρ)	0.6		0.8		1.0	
Model	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%
ALBERT	20.6	77.9	20.6	77.9	20.6	77.9
Pro-ALBERT-MCP	48.9	47.3	48.9	47.3	48.9	47.3

Table 13: Ablation study on minimum synonym cosine similarity.

Min-Cos-Sim		0.3		0.5		0.7		0.9	
Model	Clean%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%
ALBERT	93.0	12.7	86.3	20.6	77.9	40.1	56.9	79.0	15.1
Pro-ALBERT-MCP	93.8	41.1	55.7	48.9	47.3	61.6	33.6	82.7	10.9

Table 14: Ablation study on universal sentence similarity threshold.

Sentence-Sim-Threshold		0.2		0.4		0.6		0.8		0.85	
Model	Clean%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%
ALBERT	93.0	18.0	80.7	18.0	80.7	18.0	80.7	18.1	80.5	21.9	76.5
Pro-ALBERT-MCP	93.8	47.4	48.9	47.4	48.9	47.4	48.9	47.6	48.7	49.0	47.2

Sentence-Sim-Threshold	0.875		0.9		0.925		0.95	
Model	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%	Aua%	ASR%
ALBERT	27.6	70.3	46.6	49.9	71.2	23.4	90.5	2.7
Pro-ALBERT-MCP	55.9	39.8	64.8	30.2	79.4	14.4	90.4	2.6

M.5 Ablation Study on Backbone Models

Our proposed ProAttention is a universal framework which can be applied to various attention-based models. To verify the universality of our methods, we integrate our robust attention module into various backbones and present the results in Figure 13 and Table 15. As seen in the results, our ProAttention can consistently enhance the robustness over any backbone under various attacks.

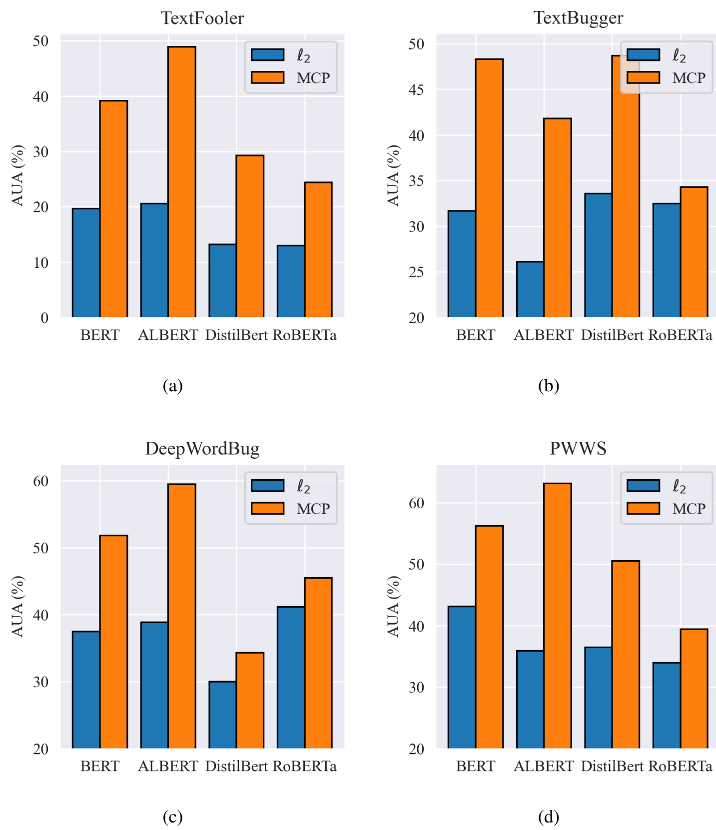


Figure 13: Accuracy under attack of different backbones.

Table 15: The results of different backbones on AGNEWS

		TextFooler			TextBugger			DeepWordBug			PWWS		
Model	Clean%	Aua%	ASR%	#Query	Aua%	ASR%	#Query	Aua%	ASR%	#Query	Aua%	ASR%	#Query
BERT	94.2	19.7	78.9	335.3	31.7	67.5	176.5	37.5	59.8	103.8	43.1	53.8	353.8
Pro-BERT-MCP	93.2	39.2	57.7	377.4	48.3	48.5	207.7	51.8	43.8	107.9	56.2	39.2	363.2
RoBERTa	93.4	13.0	86.1	301.6	32.5	64.5	180.3	41.2	55.9	105.4	34.0	63.6	345.9
Pro-RoBERTa-MCP	93.7	24.4	73.7	312.6	34.3	62.8	195.2	45.5	51.5	118.3	39.4	57.5	336.9
DistilBERT	93.5	13.2	85.9	317.4	33.6	63.4	159.1	30.0	67.9	98.0	36.5	61.0	352.4
Pro-DistilBERT-MCP	93.9	29.3	68.5	363.3	48.7	47.9	184.2	34.3	63.1	98.6	50.5	45.6	364.2
ALBERT	93.0	20.6	77.9	315.6	26.1	71.9	150.9	38.9	58.2	101.5	35.9	61.4	342.8
Pro-ALBERT-MCP	93.8	48.9	47.3	417.8	41.8	55.3	208.2	59.5	35.9	111.8	63.1	32.0	375.2

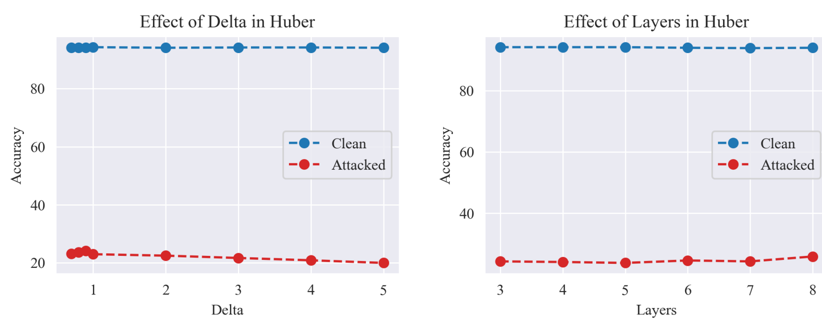
M.6 Ablation Study on Hyperparameters of Penalties

M.6.1 Ablation Study on Huber

For the Huber-based model, we present the ablation study on the δ and layers K of Huber loss under TextFooler in Table 16 and Figure 14. The default setting are as follows: $\delta = 0.9$ and $K = 3$, and we vary the two parameters separately to capture the trend. We can find that the performance is insensitive to δ or layers within an appropriate range.

Table 16: Ablation study on Huber on AGNEWS.

Model	Clean	Attacked
Pro-BERT-Huber $\delta = 0.6$	94.2	23.9
Pro-BERT-Huber $\delta = 0.7$	94.2	23.2
Pro-BERT-Huber $\delta = 0.8$	94.2	23.7
Pro-BERT-Huber $\delta = 0.9$	94.2	24.2
Pro-BERT-Huber $\delta = 1$	94.3	23.0
Pro-BERT-Huber $\delta = 2$	94.1	22.5
Pro-BERT-Huber $\delta = 3$	94.2	21.7
Pro-BERT-Huber $\delta = 4$	94.2	20.9
Pro-BERT-Huber $\delta = 5$	94.1	20.0
Pro-BERT-Huber $K = 3$	94.2	24.2
Pro-BERT-Huber $K = 4$	94.2	24.0
Pro-BERT-Huber $K = 5$	94.2	23.7
Pro-BERT-Huber $K = 6$	94.0	24.5
Pro-BERT-Huber $K = 7$	93.9	24.2
Pro-BERT-Huber $K = 8$	94.0	25.8



(a) Effect of δ.

(b) Effect of layers.

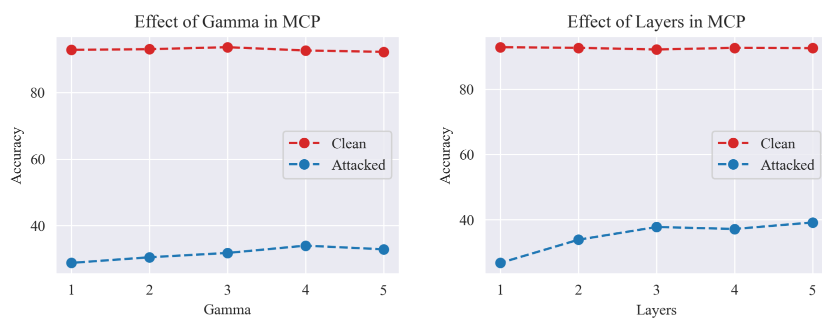
Figure 14: Ablation study on Huber

M.6.2 Ablation Study on MCP

We present the ablation study of the γ and layers K in MCP in Table 17 & Figure 15 and Table 18 & Figure 16 . From the figures, it can be concluded that appropriate γ is needed to get the best robustness. Besides, more layers can get a more precise solution for the robust objective, but we need to consider a good balance between precision and efficiency.

Table 17: Ablation on MCP on AGNEWS.

Model	Clean	TextFooler
Pro-BERT-MCP $K = 2, \gamma = 5$	93.3	32.8
Pro-BERT-MCP $K = 2, \gamma = 4$	93.7	33.9
Pro-BERT-MCP $K = 2, \gamma = 3$	93.7	31.7
Pro-BERT-MCP $K = 2, \gamma = 2$	93.1	30.4
Pro-BERT-MCP $K = 2, \gamma = 1$	92.9	28.7
Pro-BERT-MCP $K = 5, \gamma = 4$	93.6	39.2
Pro-BERT-MCP $K = 4, \gamma = 4$	93.7	37.2
Pro-BERT-MCP $K = 3, \gamma = 4$	93.2	37.8
Pro-BERT-MCP $K = 2, \gamma = 4$	93.7	33.9
Pro-BERT-MCP $K = 1, \gamma = 4$	93.9	26.8



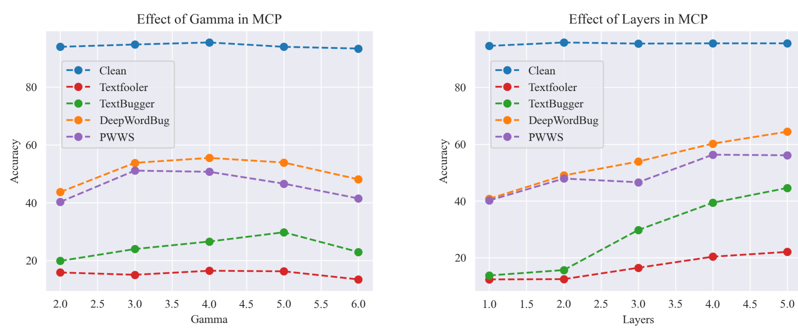
(a) Effect of γ in MCP

(b) Effect of layers in MCP

Figure 15: Ablation study on MCP on AGNEWS.

Table 18: Ablation in MCP on IMDB.

Model	Clean	TextFooler	TextBugger	DeepWordBug	PWWS
Pro-BERT-MCP $\gamma = 2.0, K = 3$	93.9	15.9	19.9	43.7	40.3
Pro-BERT-MCP $\gamma = 3.0, K = 3$	93.7	15.1	24.0	53.8	51.1
Pro-BERT-MCP $\gamma = 4.0, K = 3$	93.4	16.5	26.6	55.5	50.7
Pro-BERT-MCP $\gamma = 5.0, K = 3$	93.9	16.3	29.8	53.9	46.6
Pro-BERT-MCP $\gamma = 6.0, K = 3$	93.3	13.5	23.0	48.1	41.5
Pro-BERT-MCP $K = 1$	93.6	12.4	13.8	40.8	40.2
Pro-BERT-MCP $K = 2$	93.8	12.5	15.7	49.0	47.9
Pro-BERT-MCP $K = 3$	93.4	16.5	29.8	53.9	46.6
Pro-BERT-MCP $K = 4$	93.5	20.4	39.4	60.2	56.3
Pro-BERT-MCP $K = 5$	93.5	22.1	44.6	63.3	56.1



(a) Effect of γ in MCP

(b) Effect of layers in MCP

Figure 16: Ablation study on MCP on IMDB.

N ADDITIONAL EXPERIMENTS ON LLMS

N.1 EXPERIMENTS ON T5

N.1.1 MAIN RESULT

We compare the backbone model T5 with its robust version based on ℓ_1, MCP and Huber loss in Figure 17 and Table 19. The experiments are conducted on SST2 under TextFooler. As shown in the figure, Pro-T5 (Huber or ℓ_1) can slightly improve the robustness of backbone T5. Moreover, Pro-T5 (MCP) significantly outperform other baselines, especially under the large attack budgets.

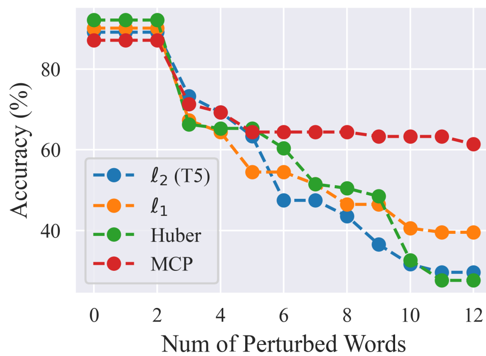


Figure 17: Main results on T5.

Table 19: Accuary (%) under prompt attack on SST2 (TextFooler, T5)

# PERTURBED WORDS	0 (CLEAN)	1	2	3	4	5	6	7	8	9	10	11	12
T5	89.1	89.1	89.1	73.3	69.3	63.4	47.5	47.5	43.6	36.6	31.7	29.7	29.7
PRO-T5 ℓ_1 $K = 3$	90.1	90.1	90.1	67.3	64.4	54.5	54.5	51.5	46.5	46.5	40.6	39.6	39.6
PRO-T5 MCP $K = 4, \gamma = 3.0$	87.1	87.1	87.1	71.3	69.3	64.4	64.4	64.4	64.4	63.3	63.3	63.3	61.4
PRO-T5 HUBER $K = 3, \delta = 3$	95.0	95.0	95.0	64.4	58.4	56.4	52.5	52.5	52.5	43.6	33.7	31.7	31.7
PRO-T5 HUBER-MCP $K = 3, \delta = 9, \gamma = 15$	89.1	89.1	89.1	62.4	62.4	57.4	55.4	55.4	55.4	55.4	55.4	55.4	54.5

N.1.2 Ablation on Huber

We present the ablation study on δ in Huber on SST2 in Figure 18 and Table 20. We fix the number of layers as 3 and vary the values of δ from 3.0 to 9.0. The Huber-based model with $\delta = 4.0$ perform best among all the selected parameters.

Figure 18: Ablation study on Huber on T5

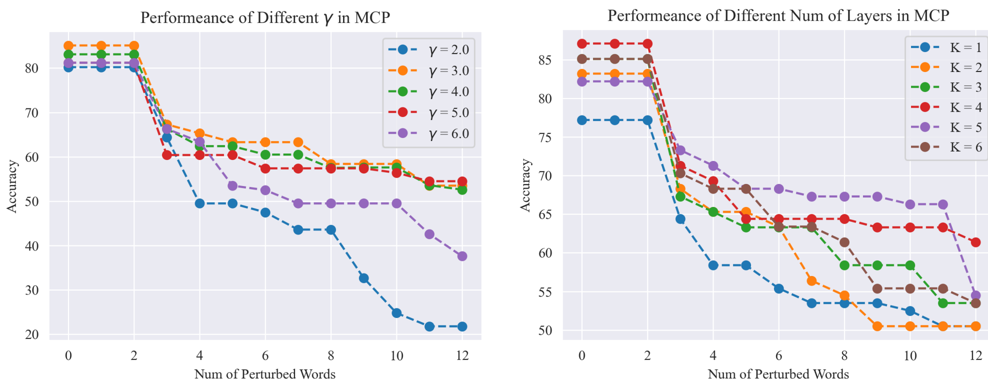
Table 20: Ablation study in Huber on SST2.

# Perturbed words	0 (Clean)	1	2	3	4	5	6	7	8	9	10	11	12
Pro-T5 Huber $K = 3, \delta = 3$	95.0	95.0	95.0	64.4	58.4	56.4	52.5	52.5	52.5	43.6	33.7	31.7	31.7
Pro-T5 Huber $K = 3, \delta = 4$	95.0	95.0	95.0	65.3	65.3	65.3	64.4	63.4	58.4	58.4	50.5	34.7	31.7
Pro-T5 Huber $K = 3, \delta = 5$	92.1	92.1	92.1	66.3	65.3	65.3	60.4	51.5	50.5	48.5	32.7	27.7	27.7
Pro-T5 Huber $K = 3, \delta = 6$	93.1	93.1	93.1	69.3	63.4	63.4	34.7	34.7	34.7	31.7	26.7	20.8	20.8
Pro-T5 Huber $K = 3, \delta = 7$	93.1	93.1	93.1	72.3	72.3	64.4	52.5	32.7	32.7	30.7	23.8	23.8	23.8
Pro-T5 Huber $K = 3, \delta = 8$	93.1	93.1	93.1	75.2	41.6	41.6	33.7	33.7	30.7	30.7	27.7	27.7	25.7
Pro-T5 Huber $K = 3, \delta = 9$	92.1	92.1	92.1	75.2	75.2	43.6	32.7	32.7	29.7	29.7	26.7	26.7	26.7

N.1.3 Ablation on MCP of T5

We present the ablation studies in MCP on SST2 in Figure 19 and Table 21. The default settings are as follows: $K = 3$ and $\gamma = 3.0$. We can make the observations as follows: (1) For γ, the optimal value falls into the range of (3,5). (2) For number of layers, the best value can be chosen as 4 or 5.

Figure 19: Ablation study on MCP of T5



(a) Effect of γ in MCP

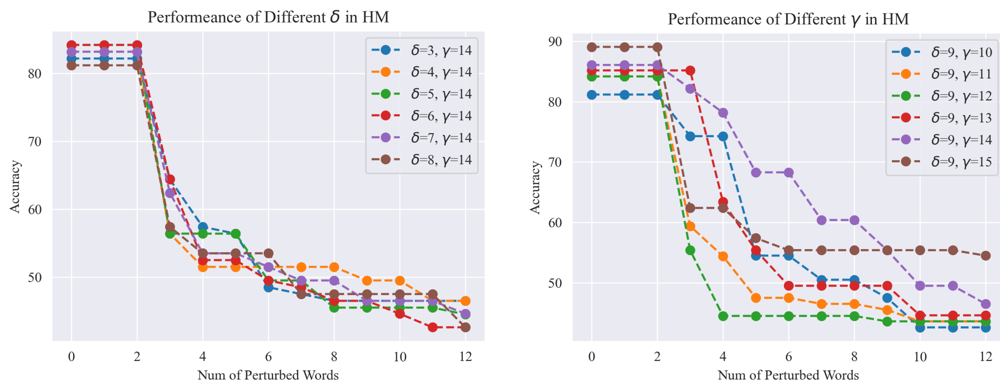
(b) Effect of K in MCP

Table 21: Ablation study on MCP on SST2

# Perturbed words	0 (Clean)	1	2	3	4	5	6	7	8	9	10	11	12
Pro-T5 MCP $K = 3, \gamma = 2.0$	80.2	80.2	80.2	64.4	49.5	49.5	47.5	43.6	43.6	32.7	24.8	21.8	21.8
Pro-T5 MCP $K = 3, \gamma = 3.0$	85.1	85.1	85.1	67.3	65.3	63.3	63.3	63.3	58.4	58.4	58.4	53.5	53.5
Pro-T5 MCP $K = 3, \gamma = 4.0$	83.1	83.1	83.1	66.3	62.4	62.4	60.5	60.5	57.5	57.6	57.6	53.6	52.6
Pro-T5 MCP $K = 3, \gamma = 5.0$	81.2	81.2	81.2	60.4	60.4	60.4	57.4	57.4	57.4	57.4	56.4	54.5	54.5
Pro-T5 MCP $K = 3, \gamma = 6.0$	81.2	81.2	81.2	66.3	63.4	53.5	52.5	49.5	49.5	49.5	49.5	42.6	37.6
Pro-T5 MCP $K = 1, \gamma = 3.0$	77.2	77.2	77.2	64.4	58.4	58.4	55.4	53.5	53.5	53.5	52.5	50.5	50.5
Pro-T5 MCP $K = 2, \gamma = 3.0$	83.2	83.2	83.2	68.3	65.3	65.3	63.4	56.4	54.5	50.5	50.5	50.5	50.5
Pro-T5 MCP $K = 3, \gamma = 3.0$	85.1	85.1	85.1	67.3	65.3	63.3	63.3	63.3	58.4	58.4	58.4	53.5	53.5
Pro-T5 MCP $K = 4, \gamma = 3.0$	87.1	87.1	87.1	71.3	69.3	64.4	64.4	64.4	64.4	63.3	63.3	63.3	61.4
Pro-T5 MCP $K = 5, \gamma = 3.0$	82.2	82.2	82.2	73.3	71.3	68.3	68.3	67.3	67.3	67.3	66.3	66.3	54.5
Pro-T5 MCP $K = 6, \gamma = 3.0$	85.1	85.1	85.1	70.3	68.3	68.3	63.4	63.4	61.4	55.4	55.4	55.4	53.5

N.1.4 Ablation on Huber-MCP

We present the ablation study on Huber-MCP on SST2 in Figure 20 and Table 22. The results show that Pro-T5 (Huber-MCP) is insensitive to δ and get better robustness at $\gamma = 14$ or 15.



(c) Effect of δ in Huber-MCP

(d) Effect of γ in Huber-MCP

Figure 20: Ablation studies on Huber-MCP

Table 22: Ablation study on Huber-MCP on SST2

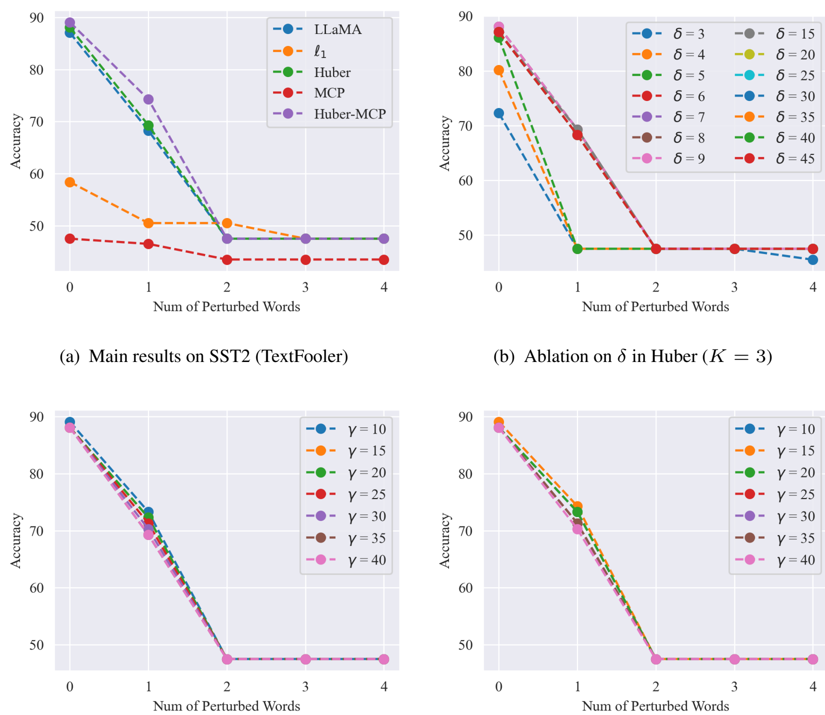
# Perturbed words	0 (Clean)	1	2	3	4	5	6	7	8	9	10	11	12
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 10$	81.2	81.2	81.2	74.3	74.3	54.5	54.5	50.5	50.5	47.5	42.6	42.6	42.6
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 11$	84.2	84.2	84.2	59.4	54.4	47.5	47.5	46.5	46.5	45.5	43.6	43.6	43.6
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 12$	84.2	84.2	84.2	55.4	44.5	44.5	44.5	44.5	44.5	43.6	43.6	43.6	43.6
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 13$	85.2	85.2	85.2	85.2	63.4	55.4	49.5	49.5	49.5	49.5	44.6	44.6	44.6
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 14$	86.1	86.1	86.1	82.2	78.2	68.3	68.3	60.4	60.4	55.4	49.5	49.5	46.5
Pro-T5 Huber-MCP $K = 3, \delta = 9, \gamma = 15$	89.1	89.1	89.1	62.4	62.4	57.4	55.4	55.4	55.4	55.4	55.4	55.4	54.5
Pro-T5 Huber-MCP $K = 3, \delta = 3, \gamma = 14$	82.2	82.2	82.2	64.4	57.4	56.4	48.5	47.5	46.5	46.5	46.5	46.5	46.5
Pro-T5 Huber-MCP $K = 3, \delta = 4, \gamma = 14$	83.2	83.2	83.2	56.4	51.5	51.5	51.5	51.5	51.5	49.5	49.5	46.5	46.5
Pro-T5 Huber-MCP $K = 3, \delta = 5, \gamma = 14$	84.2	84.2	84.2	56.4	56.4	56.4	49.5	49.5	45.5	45.5	45.5	45.5	44.5
Pro-T5 Huber-MCP $K = 3, \delta = 6, \gamma = 14$	84.2	84.2	84.2	64.4	52.5	52.5	49.5	48.5	46.5	46.5	44.6	42.6	42.6
Pro-T5 Huber-MCP $K = 3, \delta = 7, \gamma = 14$	83.2	83.2	83.2	62.4	53.5	53.5	51.5	49.5	49.5	46.5	46.5	46.5	44.6
Pro-T5 Huber-MCP $K = 3, \delta = 8, \gamma = 14$	81.2	81.2	81.2	57.4	53.5	53.5	53.5	47.5	47.5	47.5	47.5	47.5	42.6

N.2 Experiments on LLaMA

For LLaMA, we can observe an intriguing phenomenon that differs from the T5 case: the ℓ_1 and MCP-based methods sacrifice too much accuracy while Huber method can keep decent performance under small budgets. This reason is that in the small range region, ℓ_1 and MCP utilize a linear or concave functions while Huber can recover the ℓ_2 function. Inspired by the characteristics of these functions, we combine the properties of Huber and MCP, and construct a new function which we refer to Huber-MCP[1] . The detailed formulation and derived robust attention layers are available in Appendix G. As indicated by the following curves, Huber-MCP and Huber-based models exhibits better robustness than other methods while preserving the good clean performance.

N.2.1 TextFooler

We present the results of textual entailment on SST2 under TextFooler in Figure 21. We can observe that ℓ_1 and MCP-based methods sacrifice the performance because of the estimation bias. Pro-LLaMA (Huber-MCP) shows slight improvement over other models.



(a) Main results on SST2 (TextFooler)

(b) Ablation on δ in Huber ($K = 3$)

(c) Ablation on γ in Huber-MCP ($K = 1, \delta = 9$)

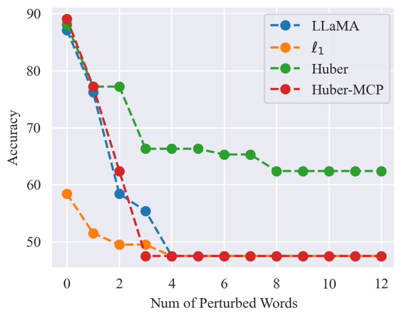
(d) Ablation on γ in Huber-MCP ($K = 3, \delta = 9$)

Figure 21: LLaMA (Textfooler)

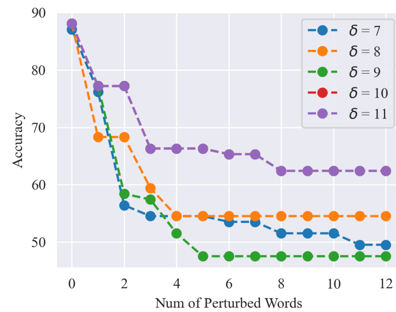
[1]Empirical penalty selection strategy: For small or medium-sized models such as BERT (110M) and ViT (86M), MCP-based models exhibit superior robustness while nearly not sacrificing the clean performance. Moreover, MCP-based models are easy to tune with only one parameter γ. For large models like LLaMA (7B) and Vicuna (7B), it is necessary to choose Huber and Huber-MCP to recover the original ℓ_2 penalty within the low-value region in case of clean performance drop.

N.2.2 TEXTBUGGER

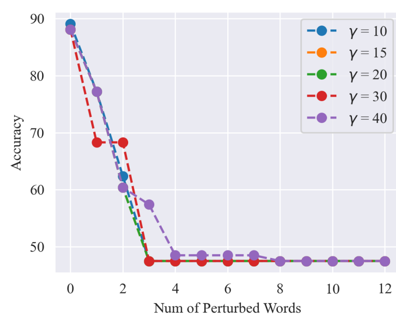
We present the results of textual entailment on SST2 under TextBugger in Figure 22. In this case, ℓ_1-based model show a catastrophic performance drop while Pro-LLaMA (Huber) outperforms other baselines with a sinificant margin.



(a) Main results on SST2 (TextBugger)



(b) Ablation study on δ in Huber ($K = 3$)

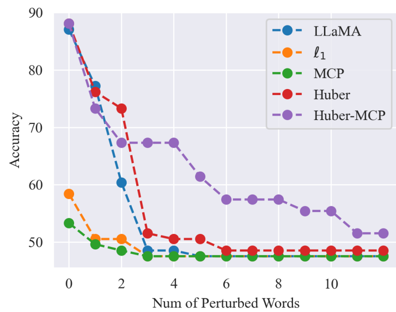


(c) Ablation study on γ in Huber-MCP ($K = 3, \delta = 9$)

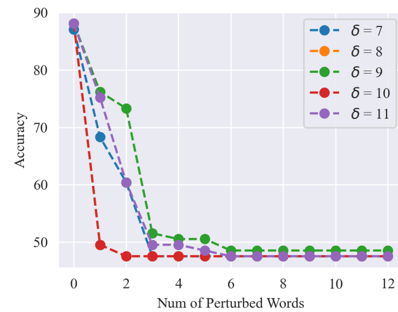
Figure 22: LLaMA (TextBugger)

N.2.3 DEEPWORDBUG

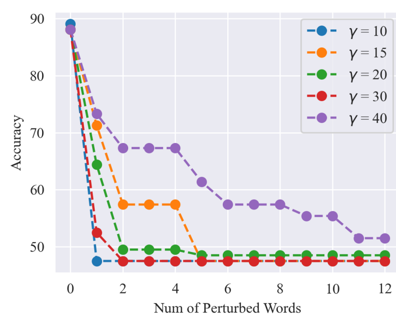
We present the results of textual entailment on SST2 under DeepWordBug in Figure 23. The experiment shows the similar phenomenon that ℓ_1 and MCP-based models sacrifice too much performance. Additionally, Pro-LLaMA (Huber-MCP) significantly outperforms other methods.



(a) Main results on SST2 (DeepWordBug)



(b) Ablation study on δ in Huber ($K = 3$)



(c) Ablation study on γ in Huber-MCP ($\delta = 9$, $L = 3$)

Figure 23: LLaMA (DeepWordBug)

O Additional Experiments on Jailbreak

O.1 Transfer Jailbreak

We provide the results of transfer jailbreak in Table 23, Table 24, and Table 25. As SmoothLLM exhibits excellent performance to defend the jaibreaking attack, we also combine the random smoothing with the backbone models and our methods to further validate the effectiveness of our method. As shown in the results, when $q = 0$ (without random smoothing), by simply plugging our ProAttention into the Vicuna, the robustness can be improved by a significant margin. Our plug-and-play method can even be comparable to the randomly smoothed models which require multiple operations including random perturbations, votings and aggregations.

Table 23: Vicuna: ASRs of JailBreak with **Swap** random smoothing on Behaviours.

Smooth q	0	1	3	5
Vicuna	91.8	71.8	20.9	0.9
Pro-Vicuna-Huber $\delta = 0.1$	1.8	0.9	0.9	0.9
Pro-Vicuna-Huber $\delta = 0.2$	8.2	1.8	0.9	0.9
Pro-Vicuna-Huber $\delta = 0.3$	21.8	2.7	0.9	0.9
Pro-Vicuna-Huber $\delta = 0.4$	30.9	21.8	0.9	0.9
Pro-Vicuna-Huber $\delta = 0.5$	36.4	23.6	0.9	0.9
Pro-Vicuna-Huber $\delta = 0.8$	61.8	40.0	0.9	0.9
Pro-Vicuna-Huber $\delta = 1.0$	70.0	53.6	11.8	0.9
Pro-Vicuna-Huber $\delta = 1.5$	74.5	60.0	16.4	1.8
Pro-Vicuna-Huber $\delta = 2.0$	82.7	73.6	21.8	1.8
Pro-Vicuna-Huber $\delta = 3.0$	90.0	74.5	31.8	7.3

Table 24: Vicuna: ASRs of JailBreak with **Insert** random smoothing on Behaviours.

Smooth q	0	1	3	5	10	15
vicuna	91.8	79.1	44.5	10.9	4.5	1.8
Pro-vicuna-Huber $\delta = 0.1$	1.8	0.9	0.9	0.9	0.9	0.9

Table 25: Vicuna: ASRs of JailBreak with **Patch** random smoothing on Behaviours.

Smooth q	0	1	3	5	10	15
vicuna	91.8	71.8	57.3	39.1	21.8	14.5
Pro-vicuna-Huber $\delta = 0.1$	1.8	0.9	0.9	0.9	0.9	0.9

O.2 Adaptive Jailbreak

Besides transfer jailbreak, we also provide the results of adaptive jailbreak in Table 26 and Figure 24. The results show that our methdod can consistently enhance the backbone model across various numbers of attack queries.

Table 26: vicuna: ASRs of **Adaptive** JailBreak on Behaviours

Num of Attack Queries	12	13	14	15	16	17	18	19	20
vicuna	61.4	65.2	71.5	75.8	78.7	82.6	84.1	86.5	87.4
Pro-Vicuna (Best)	50.7	55.9	60.8	64.3	67.4	70.5	74.0	77.7	78.6
Pro-vicuna-Huber $\delta = 0.3$	60.2	60.2	65.0	70.9	72.8	75.7	77.7	77.7	78.6
Pro-vicuna-Huber $\delta = 0.5$	60.8	61.8	62.7	66.7	67.6	71.6	78.4	81.4	82.4
Pro-vicuna-Huber $\delta = 0.7$	50.7	55.9	60.8	64.3	67.4	70.5	74.0	79.7	82.4

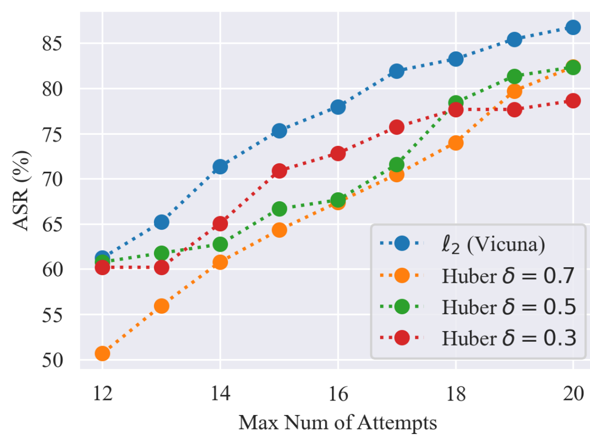


Figure 24: Adaptive JailBreak

P ADDITIONAL EXPERIMENTS ON VIT

P.1 MAIN RESULTS

The main results of FGSM on ViT are presented in Table 27. We can conclude that our Pro-ViT (MCP) outperforms other methods across various budgets.

Table 27: Adversarial robustness on CIFAR-10 (FGSM)

.

MODEL \ BUDGET	0 (CLEAN)	8/255	4/255	1/255
VIT	98.74	35.05	39.04	60.43
PRO-VIT-L1	98.46	42.41	46.33	61.99
PRO-VIT-HUBER	98.76	37.47	41.93	62.72
PRO-VIT-MCP (OURS)	98.40	54.10	60.38	75.85

P.2 ABLATION STUDY.

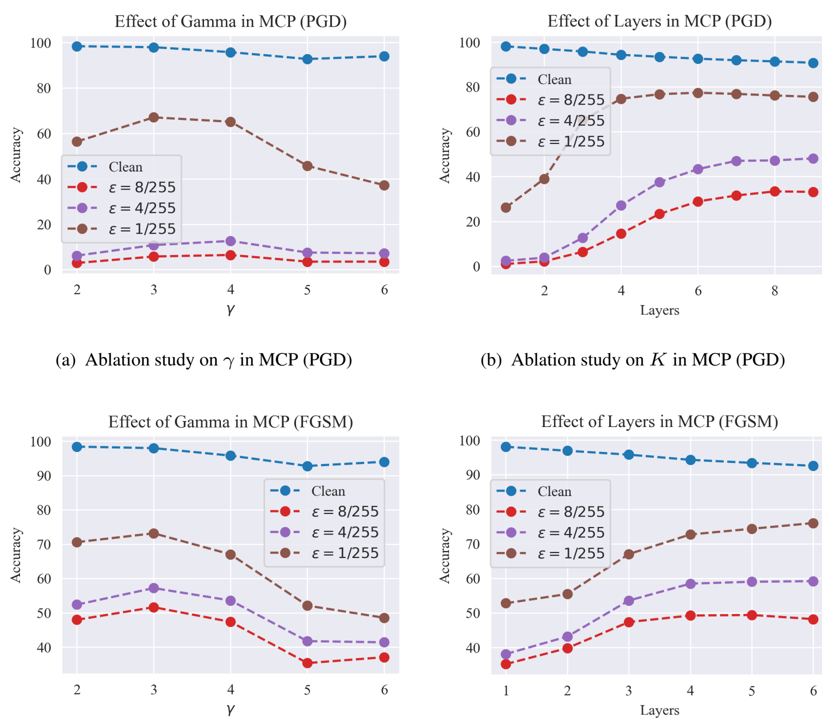
The ablation study of PGD and FGSM are provided in Table 28, Table 29 and Figure 25. As shown in the results, the optimal γ of MCP fall into the range of (3,4). The robust estimators with more layers show the better robustness while slightly sacrifice the clean performance.

Table 28: Ablation: CIFAR-10 (PGD)

MODEL \ BUDGET	0 (CLEAN)	8/255	4/255	1/255
PRO-VIT-HUBER $K = 3, \delta = 1$	98.43	0.09	0.82	28.38
PRO-VIT-HUBER $K = 3, \delta = 3$	98.56	0.07	1.36	31.04
PRO-VIT-HUBER $K = 3, \delta = 5$	98.56	0.09	1.57	34.9
PRO-VIT-HUBER $K = 3, \delta = 7$	98.76	0.15	1.72	34.89
PRO-VIT-HUBER $K = 3, \delta = 9$	98.75	0.18	1.86	34.98
PRO-VIT-MCP $K = 1, \gamma = 4$	98.07	1.03	2.43	26.16
PRO-VIT-MCP $K = 2, \gamma = 4$	96.92	2.22	3.85	39.04
PRO-VIT-MCP $K = 3, \gamma = 4$	95.79	6.47	12.65	65.15
PRO-VIT-MCP $K = 4, \gamma = 4$	94.29	14.64	27.17	74.72
PRO-VIT-MCP $K = 5, \gamma = 4$	93.43	23.34	37.56	76.75
PRO-VIT-MCP $K = 6, \gamma = 4$	92.56	28.94	43.34	77.39
PRO-VIT-MCP $K = 7, \gamma = 4$	91.89	31.57	47.01	76.86
PRO-VIT-MCP $K = 8, \gamma = 4$	91.36	33.4	47.22	76.16
PRO-VIT-MCP $K = 9, \gamma = 4$	90.76	33.17	48.11	75.56
PRO-VIT-MCP $K = 3, \gamma = 2$	98.4	2.95	6.19	56.41
PRO-VIT-MCP $K = 3, \gamma = 3$	97.97	5.8	10.83	67.07
PRO-VIT-MCP $K = 3, \gamma = 4$	95.79	6.47	12.65	65.15
PRO-VIT-MCP $K = 3, \gamma = 5$	92.77	3.53	7.54	45.70
PRO-VIT-MCP $K = 3, \gamma = 6$	94.0	3.54	7.22	37.22

Table 29: Ablation: CIFAR-10 (FGSM)

MODEL \ BUDGET	0 (CLEAN)	8/255	4/255	1/255
PRO-VIT-HUBER $K = 3, \delta = 1$	98.43	36.23	40.57	58.84
PRO-VIT-HUBER $K = 3, \delta = 3$	98.56	36.99	40.88	60.54
PRO-VIT-HUBER $K = 3, \delta = 5$	98.65	37.47	41.93	62.72
PRO-VIT-HUBER $K = 3, \delta = 7$	98.76	36.55	41.02	62.20
PRO-VIT-HUBER $K = 3, \delta = 9$	98.75	35.75	40.39	61.38
PRO-VIT-MCP $K = 1, \gamma = 4$	98.07	35.22	38.12	52.82
PRO-VIT-MCP $K = 2, \gamma = 4$	96.92	39.84	43.19	55.49
PRO-VIT-MCP $K = 3, \gamma = 4$	95.79	47.38	53.6	67.03
PRO-VIT-MCP $K = 4, \gamma = 4$	94.29	49.26	58.49	72.71
PRO-VIT-MCP $K = 5, \gamma = 4$	93.42	49.42	59.03	74.35
PRO-VIT-MCP $K = 6, \gamma = 4$	92.56	48.23	59.21	76.01
PRO-VIT-MCP $K = 3, \gamma = 2$	98.4	47.98	52.39	70.59
PRO-VIT-MCP $K = 3, \gamma = 3$	97.97	51.64	57.21	73.16
PRO-VIT-MCP $K = 3, \gamma = 4$	95.79	47.38	53.6	67.03
PRO-VIT-MCP $K = 3, \gamma = 5$	92.77	35.37	41.76	52.10
PRO-VIT-MCP $K = 3, \gamma = 6$	94.0	37.08	41.42	48.56
PRO-VIT-MCP $K = 3, \gamma = 3$	97.97	51.64	57.21	73.16
PRO-VIT-MCP $K = 4, \gamma = 3$	97.76	54.1	59.66	75.30
PRO-VIT-MCP $K = 5, \gamma = 3$	97.75	53.29	60.08	75.85
PRO-VIT-MCP $K = 6, \gamma = 3$	97.74	52.37	60.38	75.70



(a) Ablation study on γ in MCP (PGD)

(b) Ablation study on K in MCP (PGD)

(c) Ablation study on γ in MCP (FGSM)

(d) Ablation study on L in MCP (FGSM)

Figure 25: Ablation study in MCP

Q ADDITIONAL EXPERIMENTS IN GAT

The main results on Citeseer and the ablation study on Cora-ML are presented in Table 30 and Table 31. From the results, we can make the following conclusions: (1) Our Pro-GAT outperform other methods significantly. Under the larger budgets, the outliers introduced by the adversarial attack will enlarge the bias effect of the estimation. In this scenario, our MCP function can mitigate or even remove the effect of outlying values in the large-value region. (2) The parameter γ provide an implication for the robustness. For small budget, the models with large γ perform well since it is more similar the original attention. While for large budget, the models with small γ offer better robustness cause it can mitigate the bias introduced by the outliers.

Table 30: Results on Citeseer.

MODEL \ BUDGET	0%	5%	10%	20%	30%	40%
GCN	$\mathbf{74.8 \pm 1.2}$	66.1 ± 1.0	60.9 ± 0.8	53.0 ± 1.0	47.0 ± 0.8	41.2 ± 1.1
GNNGUARD	72.4 ± 1.1	65.6 ± 0.9	61.8 ± 1.4	55.6 ± 1.4	51.0 ± 1.3	47.3 ± 1.3
RGCN	74.4 ± 1.0	66.0 ± 0.8	60.6 ± 0.9	52.5 ± 0.8	46.1 ± 0.9	40.2 ± 1.0
GRAND	$\mathbf{74.8 \pm 0.6}$	66.6 ± 0.7	61.8 ± 0.7	53.6 ± 1.1	47.4 ± 1.2	42.2 ± 0.9
PROGNN	74.2 ± 1.3	65.6 ± 1.1	60.3 ± 1.1	52.7 ± 1.4	46.2 ± 0.9	40.8 ± 0.6
JACCARD-GCN	$\mathbf{74.8 \pm 1.2}$	66.3 ± 1.2	60.9 ± 1.2	53.3 ± 0.9	46.5 ± 0.9	41.1 ± 1.0
SOFTMEDIAN	74.6 ± 0.7	68.0 ± 0.7	64.4 ± 0.9	59.3 ± 1.1	55.2 ± 2.0	51.9 ± 2.1
GAT	73.4 ± 1.2	65.4 ± 1.3	60.4 ± 1.4	52.6 ± 2.5	47.2 ± 3.4	41.2 ± 4.8
PRO-GAT (OURS)	73.4 ± 1.1	$\mathbf{68.9 \pm 1.4}$	$\mathbf{66.0 \pm 2.2}$	$\mathbf{63.0 \pm 2.4}$	$\mathbf{59.5 \pm 2.6}$	$\mathbf{57.7 \pm 2.0}$

Table 31: Ablation study on Cora-ML.

MODEL \ BUDGET	0% (CLEAN)	5%	10%	20%	30%	40%
$K=1, \gamma=1.0$	84.14 ± 0.35	78.51 ± 0.39	75.70 ± 0.45	72.06 ± 0.44	69.00 ± 0.65	66.34 ± 0.99
$K=1, \gamma=2.0$	83.70 ± 0.72	78.46 ± 0.51	75.46 ± 0.80	71.21 ± 1.32	68.39 ± 1.60	65.91 ± 2.17
$K=1, \gamma=3.0$	83.95 ± 0.77	77.93 ± 0.55	74.35 ± 0.63	69.46 ± 1.16	66.31 ± 1.73	62.87 ± 1.54
$K=1, \gamma=4.0$	84.18 ± 0.64	77.41 ± 0.64	73.97 ± 0.74	68.97 ± 0.98	65.70 ± 1.20	62.70 ± 1.42
$K=1, \gamma=5.0$	83.91 ± 1.17	77.57 ± 0.93	73.88 ± 1.29	68.98 ± 1.27	65.17 ± 1.54	62.40 ± 1.75
$K=1, \gamma=6.0$	83.91 ± 0.79	77.45 ± 0.75	73.56 ± 0.88	68.66 ± 1.21	64.80 ± 1.41	61.94 ± 2.08
$K=1, \gamma=7.0$	84.26 ± 0.54	77.77 ± 0.79	74.21 ± 0.67	68.94 ± 0.88	65.20 ± 1.30	62.31 ± 1.69
$K=3, \gamma=1.0$	82.75 ± 0.87	77.59 ± 0.95	75.04 ± 1.25	71.47 ± 1.06	68.70 ± 1.20	66.53 ± 1.24
$K=3, \gamma=2.0$	80.88 ± 3.79	75.89 ± 2.88	72.61 ± 2.39	68.71 ± 2.07	65.39 ± 2.25	62.29 ± 2.65
$K=3, \gamma=3.0$	83.04 ± 1.04	77.09 ± 1.22	73.82 ± 1.23	69.27 ± 1.45	65.71 ± 1.62	62.62 ± 2.07
$K=3, \gamma=4.0$	81.84 ± 3.57	76.37 ± 2.62	73.45 ± 2.04	68.63 ± 2.49	65.09 ± 2.56	62.42 ± 2.33
$K=3, \gamma=5.0$	83.79 ± 0.75	77.81 ± 0.85	74.58 ± 0.96	69.90 ± 1.05	66.32 ± 1.26	63.33 ± 1.66
$K=3, \gamma=6.0$	83.38 ± 1.12	77.17 ± 1.15	74.12 ± 1.28	69.27 ± 1.33	65.59 ± 1.34	62.86 ± 1.75
$K=3, \gamma=7.0$	84.57 ± 0.76	78.47 ± 0.78	75.15 ± 0.84	70.47 ± 0.96	66.91 ± 1.33	63.94 ± 1.33

R COMPLEXITY ANALYSIS.

Here we will provide a complexity analysis of the vanilla attention, our robust attention, KDE-based attention and RKDE-based attention. The related notations are $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times D}$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$. The major difference in these methods is how to derive the attention matrix, therefore we will only count the complexity of attention matrix derivation and context matrix computation.

- **Vanilla Attention.** The vanilla attention matrix in can be formulated as $\mathbf{A} = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{D}}\right)$, which costs about $N \times N \times D$. The context matrix computation requires $N \times N \times D$, so the total cost is $2 \cdot (N \times N \times D)$
- **Our ProAttention.** For our robust attention, we need to firstly compute the original matrix $\mathbf{A}$ ($N \times N \times D$). Then we need to compute weight $\mathbf{W}^{(k)}$ based on the pairwise distance between the $\mathbf{V}$ and current estimator $\mathbf{Z}^{(k)}$ ($N \times N \times D$). Finally we need to update the estimator by $\mathbf{Z}^{(k+1)} = (\mathbf{A} \odot \mathbf{W}^{(k)})\mathbf{V}$ ($N \times N \times D$). The context matrix is calculated through the iterations. Therefore, the total cost will be $(1 + 2K) \cdot N \times N \times D$. As stated in the ablation study in Section D.2.2, our Newton-IRLS in ProAttention can converge efficiently within 3 steps ($K \leq 3$). Therefore, our ProAttention is still effeicient.

  Kernel Density Estimation (KDE) Attention. For KDE-based attention, the attention matrix is computed based on the pairwise ditance between the $\mathbf{K}$ and $\mathbf{Q}$, which cost $N \times N \times D$. The context matrix computation requires $N \times N \times D$, so the total cost is $2 \cdot (N \times N \times D)$.
- **Robust Kernel Density Estimation (RKDE) Attention.** For the RKDE-based attention, we need to perform the following operations. Firstly, we need to compute the basic matrix ad KDE attention matrix $\mathbf{A}$ ($N \times N \times D$). Then we need to compute pairwise distance $\mathbf{D}_{\mathbf{K}}^{(k)}$ for all the key pairs ($N \times N \times D$) and update the weight $\mathbf{W}_{\mathbf{K}}^{(k)}$ based on $\mathbf{W}_{\mathbf{K}}^{(k-1)}$ and $\mathbf{D}_{\mathbf{K}}^{(k)}$ ($N \times N \times N$). Similarly, we need calculate the pairwise distance $\mathbf{D}_{\mathbf{KV}}^{(k)}$ and update the weight $\mathbf{W}_{\mathbf{KV}}^{(k)}$ for concatenated key and value, which costs $N \times N \times 2D$ and $N \times N \times N$, repectively. The context matrix computation requires $N \times N \times D$. Therefore, the total cost will be $(2 + 3K) \cdot N \times N \times D + 2K \cdot N \times N \times N$.