Adapting Toxicity Detection from Comments to Conversations with PEFT and Knowledge Distillation

Xiao(Mike) Fang
x9fang@ucsd.edu

1 Introduction

Toxicity detection, or toxicity classification, is a common application in text classification. Our project focuses on improving task adaptation efficiency for toxicity detection with Parameter-Efficient Fine-Tuning (PEFT) and Knowledge Distillation (KD). Our base model is a pre-trained BERT classifier from Hugging Face, **Unitary/Toxic-BERT**, which is able to detect the toxicity of short comments. We aimed to adapt its capability to classifying longer conversational inputs. By fine-tuning it on the **lmsys/Toxic-Chat** dataset with different strategies, our work concentrate on the following aspects:

- **Solving data imbalance** (Done): While the **lmsys/Toxic-Chat** dataset is highly imbalanced, with 92.5% toxic samples, we performed balanced down sampling and conducted fine-tuning experiments on both the imbalanced and sampled datasets.
- **Incorporating external knowledge into fine-tuning** (Done): Along with the inputs, we utilized data from the OpenAI Moderation API, a list of Moderation scores of different toxic categories. By designing a combined loss function, we incorporated knowledge distillation into the fine-tuning objective by Multi-Target training and tested whether KD enriches the learning.
- **Introduce PEFT to different fine-tuning strategies** (Done): We conducted a comparative analysis of full-parameter and LoRA-based fine-tuning methods on both Single-Target and Multi-Target training, evaluating their training convergence speed, parameter efficiency, and toxicity detection performance.

2 Related Work

Toxicity detection has garnered significant attention in NLP due to its practical use case of moderating online content. In this section, we will review the foundational components in modern toxicity detection approaches that underpin our work, including the use of Bidirectional Encoder Representations from Transformers (BERT), parameter-efficient fine-tuning (PEFT) methods, knowledge distillation (KD), and multi-target optimization strategies.

2.1 BERT in Toxicity Detection

BERT-based models have proven effective for toxicity classification tasks due to their ability to leverage contextual embeddings. Unlike earlier methods that relied on static word embeddings such as Word2Vec or GloVe, BERT incorporates bidirectional context, enabling it to disambiguate subtle expressions of toxicity. For instance, prior studies have shown that BERT-based classifiers, such as Unitary Toxic BERT, perform well on datasets containing short toxic comments by capturing nuanced relationships between tokens and context (Kenton and Toutanova, 2019). However, adapting BERT to handle longer conversational inputs remains a challenging task, particularly when addressing domain-specific language variations and semantic complexity.

2.2 Parameter Efficient Fine Tuning

Traditional fine-tuning of pre-trained language models often involves updating all parameters, which is computationally expensive and storage-intensive. PEFT techniques such as LoRA (Hu et al., 2021), P-Tuning (Liu et al., 2021), and IA3 (Liu et al., 2022) address these challenges by focusing on lightweight parameter updates. LoRA introduces low-rank adaptations to the

weight matrices, significantly reducing memory requirements while maintaining competitive performance. Similarly, P-Tuning and IA3 optimize task-specific performance by tuning the numerical prefixes in the input space and modifies intermediate activations. These methods have demonstrated potential in various NLP tasks but require further exploration in toxicity detection scenario.

2.3 Teacher Knowledge Enriched Learning

Knowledge distillation (KD) has been widely employed to transfer knowledge from a larger, well-trained teacher model to a smaller student model (Hinton, 2015). In the context of fine-tuning, KD has been shown to enhance model generalization by incorporating additional supervision signals from the teacher. Recent studies highlight the effectiveness of KD in augmenting pre-trained models for specialized tasks, particularly when training data is limited or imbalanced (Jiao et al., 2019). By combining teacher-student learning paradigms with PEFT methods, it is possible to achieve efficient task-specific adaptation without sacrificing performance.

2.4 Multi-Target Optimization Strategies

Multi-target optimization involves designing models and objectives that balance competing learning goals. Techniques such as Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) have been employed to improve robustness while maintaining task performance. In toxicity classification, multi-target learning has been used to integrate diverse objectives, such as toxicity detection, sentiment analysis, and conversational understanding, within a unified framework (**?**). This approach allows models to capture richer linguistic phenomena while mitigating over-fitting risks in single-task settings. However, achieving effective multi-task trade-offs requires careful calibration of the optimization process, especially in resource-constrained scenarios. In our fine-tuning experiment, we used a weighting parameter σ 6.2.1, for balancing the importance of learning toxicity classification capability and fitting the distribution of moderation scores.

3 Dataset

In order to perform a convincing experiment, we chose the **lmsys/Toxic-Chat** dataset, which is a curated dataset specifically designed for toxicity detection and moderation in user-AI interaction domain. To have the dataset well-prepared for our training, we conducted various pre-processing procedures. Below, we detail the data annotation of **lmsys/Toxic-Chat**, which is already done by the author. Next, we describe our pre-processing methodology.

3.1 Data Annotation

The **lmsys/Toxic-Chat** dataset contains toxicity annotations on 10,000 user prompts collected from the Vicuna online demo. The dataset utilizes a human-AI collaborative annotation framework to guarantee the quality of annotation while maintaining a feasible annotation workload (Lin et al., 2023). We believe that **lmsys/Toxic-Chat** can be a valuable resource to drive our experiment further with more persuasiveness.

3.2 Different Dataset Versions

The dataset consists of 2 versions of user prompts annotated for toxicity and jailbreak behaviors. The latest version, `toxicchat0124`, is built on the earlier `toxicchat1123` version, incorporating model error analysis and additional annotations. In version `toxicchat0124`, errors identified in `toxicchat1123` were corrected, with fixes reviewed by two annotators. Overall, the corrections to toxicity labels were minimal (1.28% difference), and 20 additional human-annotated examples were included. Our work is based on the latest `toxicchat0124` version, which incorporates updates enhancing data reliability and providing a consistent baseline for our model evaluation.

3.3 Pre-processing

We loaded the `toxicchat0124` version dataset from Hugging Face first. To prepare the dataset for our experiments, we conducted the following pre-processing steps:

3.3.1 Data Cleaning

- *Remove non-English inputs:* Because the base model **Unitary/Toxic-BERT** is only trained on English data, we filtered out inputs that were not in English to maintain linguistic consistency between dataset and model.

- *Formalize Moderation Scores:* Moderation scores for toxicity and jailbreak labels were normalized and standardized from JSON text

Method	Dataset	LoRA	Targets	Key Parameters
Baseline	Sampled/Full	No	Single	——
LoRA-Single	Sampled/Full	Yes	Single	$r = 4$, $\alpha = 16$
Multi-Target	Sampled/Full	No	Multiple	$\sigma = 0.8$
LoRA-Multi	Sampled/Full	Yes	Multiple	$r = 4$, $\alpha = 16$, $\sigma = 0.8$

Table 1: Summary of experimental configurations.

Strategy	Dataset	LoRA	Time (Sampled)	Time (Full)
Single Target	Sampled/Full	No	1 min 37 sec	15 min 35 sec
LoRA-Single	Sampled/Full	Yes	1 min 23 sec	13 min 43 sec
Multi-Target	Sampled/Full	No	1 min 40 sec	15 min 37 sec
LoRA-Multi	Sampled/Full	Yes	1 min 24 sec	14 min 01 sec

Table 2: Summary of training times for different strategies and datasets.

to columns of float to ensure a consistent representation of all the targets.

3.3.2 Data Splitting and Sampling

- *Re-split Train/Test Sets:* The dataset splits was initially set up as 1:1. We applied a 7:3 re-split to ensure a strong knowledge base in the training data.
- *Down-Sampling for Balance:* Since the dataset exhibited significant class imbalance (92.5% of examples are toxic), we performed down-sampling to balance the toxic and non-toxic examples.

Choosing the proper data version, along with these pre-processing steps, helped us built high-quality, balanced data for training and evaluation, and eliminated key issues like inconsistent annotations and class imbalance.

4 Training Environment

We tested on 2 different machine configurations, and officially run all the experiments on GPU.

- **GPU Setup:** Google Colab's T4 GPU was used for model training (585 MHz, 16GB memory).
- **CPU Setup:** GCP E2 machine, with Intel® Xeon® Gold 6253CL processor (3.1 GHz).
- **Comparison:** The GPU-based training was much faster. When using the sampled dataset for full-scale hyper-parameter tuning, the CPU took about 2 hours, whereas the T4 GPU only took 15 minutes.

5 Baseline

5.1 Single-Target Direct Training

For the baseline, we performed single-target direct training, where the model is trained solely to predict the toxicity score without any auxiliary targets or low-rank adaptation techniques.

Model Configuration

- **Base Model:** A pretrained transformer-based classifier, `bert-base-uncased`.
- **Output Dimension:** The model is reconstructed by setting `num_labels=1`, where a single output indicates the toxicity score.

Key Training Parameters

- **Learning Rate:** We set different learning rates for different dataset scales. The full dataset uses a learning rate of 1×10^{-5}, while the sampled dataset uses a learning rate of 2×10^{-5}. Experimental results showed that reducing the learning rate for the full dataset leads to more stable loss convergence.
- **Loss Function:** Binary Cross-Entropy.
- **Optimizer:** AdamW (adamw_torch).

Dataset Configuration We train and evaluate the baseline model on two dataset scales:

- **Sampled Dataset:** A balanced sampled subset of the full dataset, referred to as `toxicchat0124_sampled`, suitable for lightweight testing.
- **Full Dataset:** The complete dataset, referred to as `toxicchat0124_processed`.

Model	Accuracy		Precision		Recall		F1 Score	
	Small	**Full**	**Small**	**Full**	**Small**	**Full**	**Small**	**Full**
Baseline-Sampled	0.8993	0.9130	0.9259	0.4552	0.8681	0.8241	0.8961	0.5865
LoRA-Sampled	0.3472	0.4768	0.2442	0.0229	0.1458	0.1435	0.1826	0.0395
Baseline-Full	0.8507	0.9650	0.9810	0.8212	0.7153	0.6806	0.8273	0.7443
LoRA-Full	0.5000	0.9255	0.0000	1.0000	0.0000	0.0046	0.0000	0.0092
Multi-Target-Sampled	0.9097	0.9241	0.9470	0.4958	0.8681	0.8287	0.9058	0.6205
LoRA-Multi-Sampled	0.5868	0.9327	1.0000	0.8056	0.1736	0.1343	0.2959	0.2302
Multi-Target-Full	0.5000	0.9251	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LoRA-Multi-Full	0.5000	0.9251	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 3: Performance metrics for all models across the **sampled and full datasets**.

Evaluation Performance evaluation of all experiments will be given in the Evaluation Section (7).

6 Our Approach

6.1 Single-Target Training with LoRA

Building upon the baseline, we incorporate LoRA, trying to reduce the computational costs.

LoRA Configuration

- **Rank (r):** 8. The rank of all LoRA layers.
- **Alpha (α):** 32. The scalar factor amplifying the low-rank outputs.
- **Target Modules:** `query`, `key`, and `value`, where LoRA is applied.

Comparison to Baseline LoRA freezes the majority of the base model's parameters, which significantly reduces the hardware requirements for GPU memory. However, the training time on the sampled dataset does not appear to be significantly reduced (2). Evaluation details will be given in the Evaluation Section (7).

6.2 Multi-Target Training

In this stage, we extend our approach to multi-target training by collaborate toxicity detection together with knowledge distillation. In our hypothesis, the moderation scores from Open AI Moderation API can act as a teacher model, guiding our model in capturing richer contextual knowledge beyond the primary toxicity label.

6.2.1 Multi-Target Direct Training

Multi-Target Setup

- **Number of Targets ($N\_TARGETS$):** 12. Toxicity and 11 of Moderation Scores.
- **Sigma (σ):** 0.8. A weighting parameter for balancing the importance of learning toxicity classification capability and fitting the distribution of moderation scores. $\sigma = 1$ will let model learn only toxicity classification. $\sigma = 0$ will let the model fit only the distribution of moderation scores.

Training Objectives

- **Primary Objective:** Minimize the loss from the toxicity classification task, which remains our primary evaluation focus.
- **Knowledge Distillation Objective:** Perform knowledge distillation to learn from the moderation scores provided by the OpenAI Moderation API. The goal is to capture the distribution of these scores and incorporate additional knowledge during training, enabling the model to understand more nuanced patterns beyond the primary toxicity labels.

6.2.2 Multi-Target Training with LoRA

Finally, we incorporate LoRA into the multi-target training pipeline, leveraging its efficiency across both datasets.

Compare to Multi-Target Direct Training LoRA significantly reduced the memory and computational needs. Evaluation details will be given in the Evaluation Section (7).

6.3 Summary of Experimental Setup

Table 1 summarizes the experimental configurations across all methods.

7 Evaluation

In this section, we evaluate the effectiveness of the proposed methods under multiple configurations

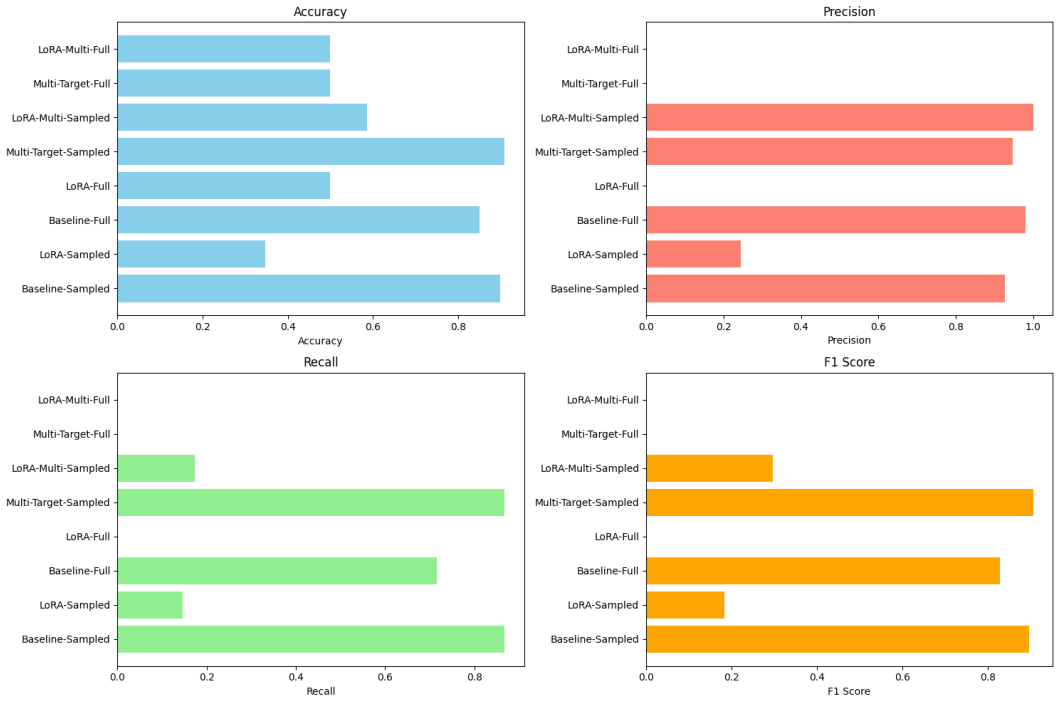

Figure 1: Performance comparison across different models on the **full test dataset**.

and fine-tuning strategies using metrics such as accuracy, precision, recall, and F1-score. The evaluation covers two datasets: the sampled (small) test set (`toxicchat0124_sampled`) and the full test set (`toxicchat0124_processed`). At the end, we also delivered a time consumption analysis.

7.1 Evaluation Metrics

Table 3 fully summarizes the evaluation results across all models for both datasets. To make it easier to understand and compare, we selected the results from full test set and displayed on figure 1.

Intuitive Observed Trends Analysis

- **Baseline**

  In general, the Baseline setting is the most stable one. Looking at Precision and Recall, we can see that there is at least one error model in the categories of LoRA or Multi-Target training, which predicts all samples as non-toxic. We will discuss this issue in the next section. However, for the Baseline setting, the test results are relatively good under both dataset sizes.

  Comparing the performance of **Baseline-Full** and **Baseline-Sampled**, from the perspective of the classifier, Baseline-Full's indicators are better than **Baseline-Sampled**. However, a closer look at the data shows that the Recall of **Baseline-Sampled** is stronger than Baseline-Full on both datasets. This is counterintuitive because there are more toxic samples in the Full Dataset. The potential explanation is that training on a larger dataset requires more epochs.

- **Effects of LoRA**

  The **LoRA-Sampled** model demonstrated reduced accuracy (34.72%) and F1-score (18.26%) compared to the baseline. This suggests that while LoRA effectively reduces parameter storage requirements, its performance may degrade on smaller datasets.

  In contrast, the **LoRA-Full** model excelled with a recall of 100% but at the cost of precision, resulting in an F1-score of 0.92%. Obviously this is an error model. After checking the model output, we found that the model classified all samples as non-toxic. This means that the model is under-fitting and the distribution of logistics has not been adjusted to the appropriate position.

- **Effects of Knowledge Distillation**

  The **Multi-Target-Sampled** model showed a strong performance on the sampled dataset,

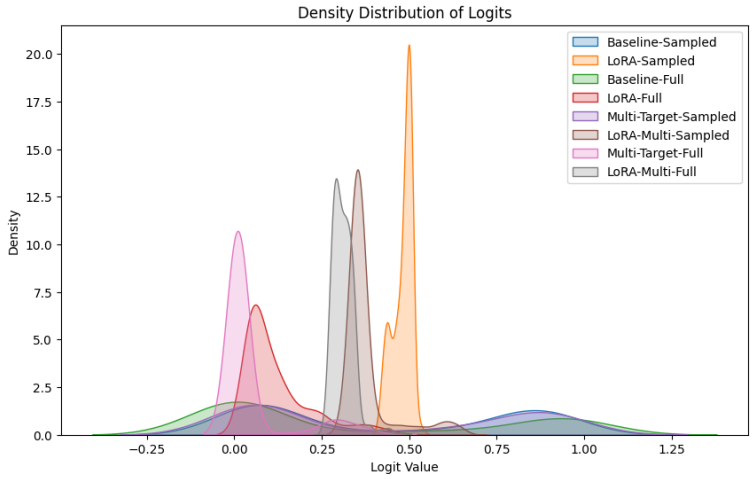

Figure 2: Logistics output after Sigmoid activation for different models on the **full test dataset**.

achieving an F1-score of 90.58%, showing the benefit of using KD-enhanced training even with limited data.

Same as **LoRA-Full** model, **Multi-Target-Full** model is also an error model. By checking the output, we found that this model classified all samples as non-toxic, which indicates under-fitting. Comparing **Multi-Target-Sampled** and **Multi-Target-Full** model, like what we seen in baseline setting, it indicates that larger datasets need more training loops.

- **Knowledge Distillation in LoRA**

  The **LoRA-Multi-Sampled** model improved over **LoRA-Sampled** setups, achieving an impressive increase of 45% in F1-score on the full dataset. What's even more amazing is that the **LoRA-Multi-Sampled** model achieves 100% Precision on the Small dataset, beating all other models. This shows the great potential of KD-enhanced training in a limited parameter tuning space.

7.2 Time Consumption Comparison

Table 2 summarizes the training times for baseline, and both Regular Training and LoRA Training on both sampled and full datasets with single and multi-target setup. LoRA-based models consistently required less time than their full-parameter counterparts, with reductions of up to 12% on the full dataset. The acceleration ability of LoRA increases as the size of the data set increases.

7.3 Overall Performance Comparison

Figure 1 provides a comparative analysis of model performance. Key takeaways include:

- Baseline models tend to offer balanced performance, indicating reliable generalization.
- LoRA models display trade-offs between training speed, storage needs versus performance. However, as the size of the dataset increases, the sacrifice in performance will gradually decrease.
- Multi-Target training, or KD-enhanced fine-tuning, has an amazing performance improvement. When using full parameter tuning, this advantage is not very obvious for one target among multiple targets, but when using LoRA, the knowledge-guided fine-tuning with the help of the teacher model will bring huge improvements.

8 Error Analysis

In this section, we examined the cause of error models and analyze patterns of common error cases of different models.

8.1 Probability Distribution Analysis

Logit density distributions for selected models are plotted in Figure 2. Key observations include:

- **LoRA Models**

  LoRA-Sampled shows higher variance, which suggesting inconsistent predictions,

especially for inputs near the decision threshold (0.5). **LoRA-Full** exhibits extreme confidence in classifying inputs as non-toxic. At the same time, the entire distribution is on the left side of 0.5, which also reveals the fact that it is under-fitting.

- **Multi-Target Models**

  Both **LoRA-Multi-Target-Full** and **Multi-Target-Full** models demonstrate the same issue as **LoRA-Full**. The explanation for the three error models is that either large dataset scale, LoRA, or Multi-Target training requires more epochs. The combination of them will need more, which leads to the under-fitting we observed.

8.2 Concrete Misclassified Examples

We manual analyzed 10 commonly misclassified examples and attempt to reveal their patterns:

- **Semantic Ambiguity:** Inputs like "Can you tell me how to implement a 3-layer CNN in PyTorch?" were flagged as toxic due to phrases like "layer" being misinterpreted in specific contexts.
- **Domain-Specific Language:** Examples containing technical jargon (e.g., "Write Kubernetes manifest for deploying WordPress") were often misclassified, particularly by LoRA-based models, highlighting gaps in domain adaptation.
- **False Positives in LoRA Models:** Many false positives were observed in **LoRA-Full**, where benign prompts were classified as toxic due to overly aggressive recall optimization.

9 Conclusion

Our project demonstrated the potential of combining PEFT and Knowledge Distillation for fine-tuning. While LoRA and Multi-Target approaches showcase computational efficiency and performance improvement, some configurations experienced under-fitting due to insufficient training epoch.

In Summary, if we want to have a model fine-tuned on limited training resources and parameter storage resources, combining LoRA and Knowledge Distillation will be a very good choice.

Our future work will focus on training the under-fitted models with additional epochs, then redo the evaluation. Self-learning thresholds to dynamically optimize decision boundaries is another direction we plan to attempt.

10 Acknowledgments

- I wrote the whole report by myself (really tiring!!!). Then I used ChatGPT to check typo and adjust formatting.
- I struggled with the layout of tables at first, because the table would always be cut off when displayed in a certain column. I asked ChatGPT for help to make it display as I wanted: breaking across two columns and staying at the top of the page.

References

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hinton, G. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2019). Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota.

Lin, Z., Wang, Z., Tong, Y., Wang, Y., Guo, Y., Wang, Y., and Shang, J. (2023). Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. *arXiv preprint arXiv:2310.17389*.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Liu, X., Ji, K., Fu, Y., Tam, W. L., Du, Z., Yang, Z., and Tang, J. (2021). P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.