ON THE RESOURCE EFFICIENCY OF LANGUAGE MODELS

A Dissertation
Presented to
The Academic Faculty

By

Rongzhi Zhang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Machine Learning
School of Computational Science and Engineering

Georgia Institute of Technology

May 2025

ON THE RESOURCE EFFICIENCY OF LANGUAGE MODELS

Thesis committee:

Dr. Chao Zhang
Computational Science and Engineering
Georgia Institute of Technology

Dr. B. Aditya Prakash
Computational Science and Engineering
Georgia Institute of Technology

Dr. Tuo Zhao
Industrial and Systems Engineering,
Computational Science and Engineering
Georgia Institute of Technology

Dr. Steve Mussmann
Computer Science
Georgia Institute of Technology

Dr. Yelong Shen
Microsoft Research
Microsoft

Date approved: April 18, 2025

There is only one heroism in the world: to see the world as it is, and to love it.

Romain Rolland

For my parents

ACKNOWLEDGMENTS

The completion of this doctoral thesis marks the culmination of a challenging yet incredibly rewarding academic journey. This work would not have been possible without the guidance, support, and encouragement from numerous individuals and organizations.

First and foremost, I am deeply grateful to my advisor, Dr. Chao Zhang, for his mentorship throughout my PhD journey. His insights into machine learning and natural language processing fundamentally shaped this research. Dr. Zhang gave me both the freedom to explore and the guidance to stay focused on important problems in resource-efficient language models. His rigorous approach to research has profoundly influenced my development as a researcher.

I thank my thesis committee members for their valuable input: Dr. B. Aditya Prakash for his expertise in data mining, Dr. Tuo Zhao for his knowledge of optimization techniques, Dr. Steve Mussmann for his perspectives on statistical machine learning, and Dr. Yelong Shen for bringing practical industry insights from Microsoft Research. Their feedback significantly improved this thesis.

My colleagues at Georgia Tech provided an intellectually stimulating environment that was crucial for this work. I would like to thank Yue Yu, Lingkai Kong, and Qingru Zhang for their valuable discussions and technical insights. I am also grateful to all members of our lab, including Rui Feng, Yuchen Zhuang, Yinghao Li, Haorui Wang, Rushi Qiang, and Changhao Li for creating a supportive and collaborative atmosphere. Our discussions, collaborations, and mutual support made the research process both productive and enjoyable.

I am grateful for the opportunities to work with outstanding researchers in industry. My sincere thanks to Dr. Yelong Shen, who not only served on my committee but also mentored me during my internship at Microsoft Azure AI. I extend my gratitude to Dr. Shuohang Wang and Dr. Lucas Liu for their guidance during my time at Microsoft. I would also like to thank Dr. Jiaming Shen, Dr. Tianqi Liu, Dr. Jialu Liu, and Dr. Michael Bendersky

for their mentorship during my research internships at Google Research. I appreciate Dr. Chenwei Zhang and Dr. Xinyang Zhang for their support during my time at Amazon Stores Foundational AI. These collaborations have significantly broadened my perspective and enhanced the practical relevance of my research.

Finally, I owe my deepest thanks to my family for their unwavering support and patience. This thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Large language models (LLMs) have revolutionized a wide range of natural language processing tasks. However, their practical utility faces resource challenges in two dimensions: **data efficiency** and **model efficiency**. For post-training, LLMs face data curation challenges where high-quality labeled data is scarce and expensive to obtain, and data utilization challenges where existing methods fail to optimize model performance on limited available data. During deployment, LLMs encounter model parameter efficiency challenges from their enormous size, and inference efficiency constraints from memory-intensive operations. This thesis addresses these resource bottlenecks through two complementary thrusts:

Thrust 1: How to enhance data efficiency in the post-training stage? LLM adaptation to specific tasks and human values remains data-intensive. Two obstacles dominate this landscape: first, the scarcity of high-quality labeled data creates bottlenecks for domain adaptation; second, existing methods fail to extract maximum value from limited training examples. I introduce PRBOOST, an interactive weakly-supervised framework that discovers high-quality labeling rules, and DORM, which optimizes preference data weights to reduce data requirements for model alignment by 10 to 40 times.

Thrust 2: How to improve model efficiency in the deployment stage? LLMs' parameter counts and memory demands restrict deployment across diverse computing environments. I tackle this through two approaches: improving knowledge distillation for more efficient models, and optimizing inference-time memory usage. PTLOSS, a perturbation-based distillation framework, enhances student model generalization by generating more accurate proxy teacher distributions. LORC progressively compresses KV cache, substantially reducing GPU memory requirements while preserving performance.

These complementary thrusts form a framework for efficient, deployable language models. The findings demonstrate that targeted resource optimization enables broader LLM deployment across diverse applications and computational environments.

CHAPTER 1
INTRODUCTION

Large language models (LLMs) have revolutionized various tasks, including text generation, question answering, complex reasoning, and decision-making. Recent breakthroughs, exemplified by models such as GPT [1], Claude [2], Gemini [3], and DeepSeek [4], have demonstrated remarkable, often human-level, capabilities previously considered beyond the reach of artificial intelligence. These successes have been made possible by enormous resources, including vast training data, billions of model parameters, and substantial computational infrastructure.

However, these resource requirements create two critical bottlenecks: data efficiency in the post-training stage and model efficiency in the deployment stage. In the post-training stage, adaptation to specific tasks or alignment with human values demands large quantities of high-quality labeled data—a process both costly and time-consuming due to the limited availability of expert annotators. Furthermore, annotations collected through crowdsourcing or generated by LLMs often contain significant noise, undermining the reliability of the post-training process.

During deployment, the use of LLMs in resource-limited environments is constrained by their enormous parameter counts and memory requirements. Transformer architectures rely on Key-Value (KV) caches whose memory usage increases linearly with sequence length and batch size, quickly becoming impractical for large-scale inference. Additionally, conventional knowledge distillation methods typically force student models to replicate the unreliable or biased output distributions of teacher models, causing a discrepancy with the ground truth and consequently leading to degraded generalization performance.

This thesis addresses these challenges through two complementary aspects: improving data efficiency for the post-training stage and enhancing model efficiency for the deployment

stage, enabling broader and more effective use of LLMs across resource-limited scenarios.

1.1 Challenges

The pursuit of resource efficiency in LLMs faces the following challenges, spanning across the post-training stage and the deployment stage.

- **Data Curation Challenges.** The scarcity of high-quality labeled data presents a fundamental obstacle to LLM post-training. Conventional supervised learning requires extensive manual annotations, which are prohibitively expensive and time-consuming to collect. Domain experts' availability is limited, while crowdsourced annotations frequently contain significant noise—up to 40% in common datasets [5]. These constraints create a bottleneck in the post-training pipeline, restricting the adaptation of LLMs to new domains and specialized applications where high-quality labeled data is sparse or costly to obtain.

- **Data Utilization Challenges.** Beyond acquisition difficulties, existing approaches fail to maximize the value extracted from available training data. Weakly supervised methods rely on labeling rules that are typically incomplete and inconsistent, and learning methods in the post-training stage often treat all data points uniformly despite their varying quality and informativeness. Current methodologies lack effective mechanisms to identify the most informative instances or differentiate between reliable and unreliable annotations. This inefficient utilization compounds the data scarcity problem, requiring larger datasets than theoretically needed.

- **Model Parameter Efficiency Challenges.** The enormous parameter counts of state-of-the-art LLMs—often ranging from billions to trillions—severely limit their deployment in diverse computing environments. While knowledge distillation provides a promising solution to reduce model size, existing methods often transfer biased or

unreliable knowledge from teacher models, resulting in student models with degraded performance and limited generalization.

- **Model Inference Efficiency Challenges.** Even when model size is manageable, inference-time computational demands—particularly memory consumption—present significant deployment barriers. Transformer architectures require extensive memory for KV caches that scale linearly with sequence length, quickly exhausting available resources during generation. Existing optimization approaches either demand prohibitive computational costs through extensive retraining or implement token management strategies that ignore crucial architectural dependencies. These limitations prevent efficient deployment of LLMs in resource-constrained environments and real-time applications.

1.2 Contributions

This thesis introduces targeted solutions to these challenges, organized into two complementary parts:

1.2.1 Part I: Data Efficiency in the Post-Training Stage

- **Interactive Rule Discovery and Boosting (chapter 2).** To improve data curation efficiency in the fine-tuning stage of pre-trained language models (PLMs), I introduce PRBOOST, an interactive weakly-supervised learning framework that iteratively discovers high-quality labeling rules through PLM prompting and human validation. PRBOOST employs a boosting-based ensemble strategy that explicitly targets high-error instances to adaptively propose new rules, continuously refining the labeling process. Evaluations on relation extraction, ontology classification, and topic classification demonstrate significant performance improvements—up to 7.1% over state-of-the-art weakly-supervised methods.

- **Dynamic Data Weights Optimization (chapter 3).** To improve data utilization efficiency in LLM alignment, I propose DORM, a two-stage approach that dynamically adjusts preference data weights. By integrating epistemic uncertainty estimation with prediction disagreement measures, DORM systematically identifies and prioritizes informative and reliable data points. A bilevel optimization framework further refines these weights, dramatically improving data efficiency and achieving 90.5% accuracy on RewardBench with 10–40× less data than conventional approaches.

1.2.2 Part II: Model Efficiency in the Deployment Stage

- **Perturbation-Based Knowledge Distillation (chapter 4).** To enhance model parameter efficiency for resource-constrained deployment environment, I develop PTLoss, a perturbation-based distillation framework that improves student model generalization. PTLoss reformulates the traditional KL-based distillation objective using a Maclaurin series expansion, introducing targeted perturbations to generate a proxy teacher distribution closer to ground truth. Theoretical analyses and empirical results across multiple datasets confirm that PTLoss significantly reduces distillation risks and outperforms conventional knowledge distillation techniques.

- **Progressive KV Cache Compression (chapter 5).** To enhance LLM inference efficiency, I present LoRC, a progressive KV cache compression strategy based on low-rank approximations of KV weight matrices. LoRC explicitly accounts for inter-layer dependencies and sensitivity, compressing KV caches without extensive retraining or task-specific tuning. Evaluations demonstrate substantial GPU memory savings across various model sizes and tasks, with minimal performance degradation.

Part I

Data Efficiency in the Post-training Stage

CHAPTER 2
IMPROVING DATA CURATION EFFICIENCY VIA RULE DISCOVERY AND BOOSTING

2.1 Introduction

Weakly-supervised learning (WSL) has recently attracted increasing attention to mitigate the label scarcity issue in many NLP tasks. In WSL, the training data are generated by weak labeling rules obtained from sources such as knowledge bases, frequent patterns, or human experts. The weak labeling rules can be matched with unlabeled data to create large-scale weak labels, allowing for training NLP models with much lower annotation cost. WSL has recently achieved promising results in many tasks including *text classification* [6, 7, 8, 9], *relation extraction* [10], and *sequence tagging* [11, 12, 13].

Despite its success, WSL is limited by two major factors: 1) the labeling rules, and 2) the static learning process. First, it is challenging to provide a comprehensive and high-quality set of labeling rules a priori. Labeling rules are often human-written [14, 15], but the process of writing labeling rules is tedious and time-consuming even for experts. A few works attempt to automatically discover labeling rules by mining labeled data [16], or enumerating predefined types. However, the pre-extracted rules are restricted to frequent patterns or predefined types, which are inadequate for training an accurate model. Second, most existing WSL methods are static and can suffer from the noise in the initial weak supervision [14, 10, 9, 8, 17]. As the labeling rule set remains fixed during model training, the initial errors can be amplified, resulting in an overfitted end model. Interactive rule discovery has been explored in two recent works [18, 19], which solicits human feedback on candidate rules to refine the rule set. Unfortunately, their rule forms are limited to simple repetitive structures such as n-grams [18], and the huge rule search space makes an enumerating-pruning pipeline

not scalable for large datasets [19].

Due to the above reasons, state-of-the-art WSL methods still underperform fully-supervised methods by significant gaps on many NLP tasks. As shown in a recent study [20], the best WSL methods fall behind the best fully-supervised methods in 15 out of 18 NLP benchmarks; and the average performance gap is 18.84% in terms of accuracy or F1 score.

To bridge the gap between weakly-supervised and fully-supervised approaches, we propose an iterative rule discovery and boosting framework, namely PRBOOST for interactive WSL. Compared to existing works on WSL and active learning, PRBOOST features three key designs:

First, we design a rule discovery module that uses rule templates for prompting pre-trained language models (PLMs). By feeding difficult instances and rule templates into PLMs, the module distills knowledge from PLMs via prompting and generates candidate rules that capture key semantics of the input instances. Compared to prior works based on *n*-grams [18], our prompt-based rule discovery is more expressive and applicable to any tasks that support prompting.

Second, we design a boosting-style ensemble strategy to iteratively target difficult instances and adaptively propose new rules. In each iteration, we reweigh data by the boosting error to enforce the rule discovery module to focus on larger-error instances. This avoids enumerating all the possible rules and implementing post-filtering for novel rules, but directly targets rule discovery on large-error instances to provide complementary information to the current model.

Third, we strategically solicit human feedback to evaluate the candidate rules. Humans are asked to judge whether a candidate rule should be accepted or abstained. The accepted high-quality rules are then used to generate new weak labels that are fed into boosted model training. As the prompt-generated rules are highly interpretable, the rule evaluation is simply a binary choice task for human experts and thus effortless. Unlike traditional active learning methods that annotate individual instances, such a rule-level annotation is more

label-efficient because the annotated rules can match large amounts of instances.

We compare our method with supervised, weakly-supervised and interactive learning baselines on four tasks: relation extraction, ontology classification, topic classification, and chemical-protein interaction prediction. The results show: 1) Our method outperforms state-of-the-art weakly-supervised baselines by up to 7.1%; 2) The rule-level annotation helps the model achieve higher model performance compared to the instance-level annotation under the same budget; 3) The machine-discovered and human-evaluated rules are of high quality, which consistently refine the weak labels and the model in each iteration.

Our key contributions are: (1) a prompt-based rule discovery framework for interactive WSL, which provides flexible rule representation while capturing subtle semantics in rule generation; (2) an iterative boosting strategy for discovering novel rules from hard instances and strengthening the model by an ensemble of complementary weak models; (3) an interpretable and easy-to-annotate interactive process for rule annotation; (4) comprehensive experiments demonstrating the effectiveness of our framework.

2.2 Related Work

Weakly-Supervised Learning. WSL has recently attracted much attention in various NLP tasks. Despite their promising performance on various tasks, manually designing the rules can be time-consuming. Moreover, the noise and incompleteness of the initial rules could be propagated in model training [20]. A few works attempt to reduce human efforts in manually designing labeling rules by discovering rules from data. For example, Snuba [16] generates heuristics based on a small labeled dataset with pre-defined rule types; TALLOR [21] and GLaRA [22] study rule expansion for NER problem based on lexical information and then select rules based on a hand-tuned threshold. However, these methods discover rules in a static way and are constrained to task-specific rule types. In contrast, our framework discovers rules iteratively from the entire unlabeled dataset, which can refine the rule set and enlarge its diversity on-the-fly.

Interactive Learning. Our work is related to *active learning* (AL) as both involve human annotators in the learning process. However, the key difference is that AL labels *instances* based on various query policies [23, 24, 25, 26, 27, 28], while our method does not annotate individual instances, but uses annotated rules to match unlabeled data. This makes our method more label-efficient in leveraging human feedback for creating large-scale labeled data. To the best of our knowledge, only a few works have studied *interactive WSL* [18, 19, 29, 30] as in our problem. However, they either use simple n-gram based rules [18, 30] that fail to capture sentence-level semantics, or suffer from a huge searching space for context-free grammar rules [19]. Unlike these works, our method uses flexible rule representations based on prompts, and also uses boosting for targeted rule discovery to avoid enumerating all possible rules and performing post-filtering for novel rules.

Language Model Prompting. Our work is also related to prompt-based learning for PLMs, which converts the original task to a cloze-style task and leverages PLMs to fill the missing information [31, 32]. Prompting has been explored in various tasks, including text classification [33, 34, 35, 36], information extraction [37, 38] and text generation [39, 40]. Recent works focus on generating better prompt templates or learning implicit prompt embeddings [41, 42, 43]. However, none of these works studied prompting for generating weak labels. Our work is orthogonal to them since we do not aim to optimize prompts for the original task, but uses prompts and PLMs as a knowledge source for rule discovery.

2.3 Preliminaries

Problem Formulation. Weakly-supervised learning (WSL) creates weak labels for model training by applying labeling rules over unlabeled instances D_u. Given an unlabeled instance $\boldsymbol{x} \in D_u$, a labeling rule $r(\cdot)$ maps $\boldsymbol{x}$ into an extended label space: $r(\boldsymbol{x}) \rightarrow y \in \mathcal{Y} \cup \{0\}$. Here $\mathcal{Y}$ is the original label set for the task, and 0 is a special label indicating $\boldsymbol{x}$ is unmatchable by r. Given a set $\mathcal{R}$ of labeling rules, we can apply each rule in $\mathcal{R}$ on unlabeled instances to create a weakly labeled dataset $\mathcal{D}'_l$.

However, the initial weak labels $\mathcal{D}'_l$ can be highly noisy and incomplete, which hinder the performance of WSL. We thus study the problem of *interactive* WSL: *how can we automatically discover more high-quality labeling rules to enhance the performance of WSL?* Besides D_u and $\mathcal{D}'_l$, we also assume access to a small set of clean labels $\mathcal{D}_l$ ($|\mathcal{D}_l| \ll |\mathcal{D}_u|$), and the task is to iteratively find a set of new rules for model improvement. In each iteration t, we assume a fixed rule annotation budget $\mathcal{B}$, *i.e.*, one can propose at most $\mathcal{B}$ candidate rules $\mathcal{R}_t = \{r_j\}_{j=1}^{\mathcal{B}}$ to human experts for deciding whether each rule should be accepted or not. The accepted rules $\mathcal{R}_t^+$ are then used to create new weakly labeled instances $\mathcal{D}'_t$. From $\mathcal{D}'_t \cup \mathcal{D}'_l$, a model $m_t : \mathcal{X} \to \mathcal{Y}$ can be trained to boost the performance of the current WSL model.

Rule Representation. Multiple rule representations have been proposed in WSL for NLP tasks. For example, *keyword-based rules* are widely used to map certain keywords to their highly correlated labels [18, 8, 7, 44]. *Regular expression* is another common rule format, which matches instances with pre-defined surface patterns [6, 9, 10]. *Logical rules* [45, 21] perform logical operations (such as conjunction $\wedge$ and negation $\neg$) over atomic rules and can thus capture higher-order compositional patterns.

We adopt a prompt-based rule representation (Section Table 2.4.1), which is flexible to encompass any existing rule representations. Our prompt-based rule relies on a rule template $\tau(\cdot)$ for the target task, which contains a `[MASK]` token to be filled by a PLM $\mathcal{M}$ along with an unlabeled instance $\boldsymbol{x}$. From the rule template τ , each candidate rule can be automatically derived by $r = g(\mathcal{M}, \tau, \boldsymbol{x})$. Such a prompt-based rule representation is highly flexible and can be applied to any NLP tasks that support prompting (see examples in Table Table 2.1).

2.4 Methodology

Overview. PRBOOST is an iterative method for interactive WSL. In each iteration, it proposes candidate rules from large-error instances, solicits human feedback on candidate rules, generates weak labels, and trains new weak models for ensembling. Figure Figure 2.1

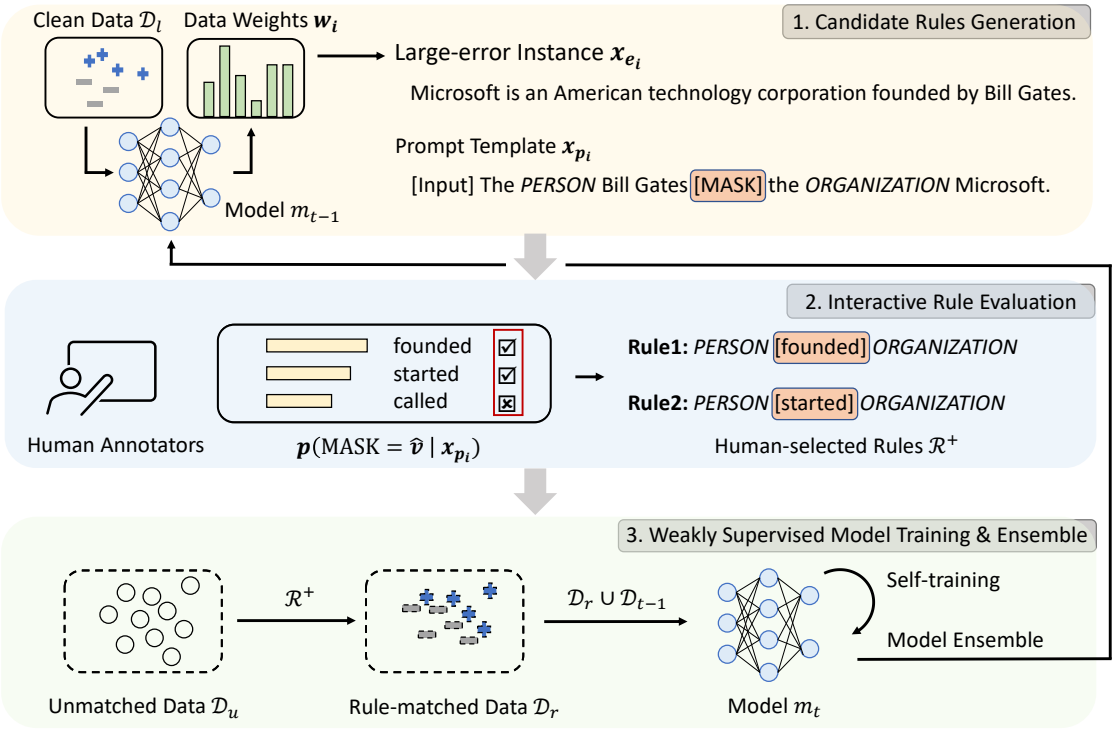


Figure 2.1: Overall framework for PRBOOST. In each iteration, PRBOOST (1) identifies large-error instances from the limited clean data and converts each large-error instance to a prompt template for prompting-based rule discovery; (2) presents candidate rules to human experts for annotation and uses accepted rules to generate new weak labels; (3) trains a new weak model with self-training and ensembles it with the previous models.

shows the process in one iteration of PRBOOST, which relies on three key components:

1. *Candidate rule generation.* This component proposes candidate rules to be evaluated by human annotators. Using the small labeled dataset $\mathcal{D}_l$, it measures the weakness of the current model by identifying large-error instances on $\mathcal{D}_l$, and proposes rules based on these instances using PLM prompting.

2. *Rule annotation and weak label creation.* This component collects human feedback to improve the weak supervision quality. It takes as input the candidate rules proposed by the previous component, and asks humans to select the high-quality ones. Then the human-selected rules $\mathcal{R}_t$ are used to generate weak labels for the unlabeled instances $\mathcal{D}_u$ in a soft-matching way.

3. *Weakly supervised model training and ensemble*. We train a new weak model m_{t+1} on the updated weakly labeled dataset $\mathcal{D}_r$. Then we self-train the weak model m_{t+1} and integrate it into the ensemble model.

2.4.1 Candidate Rule Generation

Target rule proposal on large-error instances. We design a boosting-style [46] strategy for generating prompt-based candidate rules. This strategy iteratively checks feature regimes in which the current model m_t is weak, and proposes candidate rules from such regimes. We use the small labeled dataset $\mathcal{D}_l$ to identify hard instances, i.e., where the model tends to make cumulative mistakes during iterative learning. The discovered rules can complement the current rule set $\mathcal{R}$ and refine the weak labels, so the next model m_{t+1} trained on the refined weakly labeled data can perform better in the weak regimes.

We initialize the weights of the instances in $\mathcal{D}_l$ as $w_i = 1/|\mathcal{D}_l|, i = 1, 2, \cdots, |\mathcal{D}_l|$. During the iterative model learning process, each w_i is updated as the model's weighted loss on instance $\boldsymbol{x}_i \in \mathcal{D}_l$. Specifically, in iteration $t \in \{1, \cdots, n\}$, we weigh the samples by

$$w_i \leftarrow w_i \cdot e^{\alpha_t \mathbb{I}(y_i \neq m_t(\boldsymbol{x}_i))}, i = 1, 2, \ldots, |\mathcal{D}_l|. \tag{2.1}$$

In Equation 2.1, α_t is the weight of model m_t, which will be used for both detecting hard instances and model ensembling (subsection 2.4.3). We compute α_t from the model's error rate on $\mathcal{D}_l$:

$$\alpha_t = \log \frac{1 - \mathrm{err}_t}{\mathrm{err}_t} + \log(K - 1), \tag{2.2}$$

where err_t is given by

$$\mathrm{err}_t = \sum_{i=1}^{|\mathcal{D}_l|} w_i \mathbb{I}\left(y_i \neq m_t\left(\boldsymbol{x}_i\right)\right) / \sum_{i=1}^{|\mathcal{D}_l|} w_i. \tag{2.3}$$

Intuitively, a sample $\boldsymbol{x}_i$ receives a larger weight w_i (Equation 2.1) if the model ensemble consistently make mistakes on $\boldsymbol{x}_i$. A large error is often caused by poor coverage (unlabeled instances matched by few or no rules) or dominating noise in the local feature regimes (rule-matched labels are wrong). The weights can thus guide the rule generator to target the

Table 2.1: The examples of prompt-based rules for relation extraction, ontology classification, and news topic classification. Here `[Input]` denotes the original input, `[Mask]` denotes the mask token, and $\wedge$, $\vee$ are the logical operators. We use bold words to show the ground-truth label of the original input.

Input :	Microsoft is an American technology corporation founded by Bill Gates.
Prompt :	`[Input]` The Person Bill Gates `[Mask]` the Organization Microsoft.
Rule :	{Entity Pair == (Person, Org)} $\wedge$ {`[Mask]` == *founded*} $\wedge$ $\{s_{t,j} \geq$ threshold$\}$ $\rightarrow$ **per:found**
Input :	Marvell Software Solutions Israel is a wholly owned subsidiary of Marvell Technology Group.
Prompt :	`[Input]` The Marvell Software Solutions Israel is a `[Mask]`.
Rule :	{`[Mask]` == *subsidiary* $\vee$ *corporation* $\vee$ *company*} $\wedge$ $\{s_{t,j} \geq$ threshold$\}$ $\rightarrow$ **Company**
Input :	Liverpool short of firepower for crucial encounter. Rafael Benitez must gamble with Liverpools Champions League prospects tonight but lacks the ammunition to make it a fair fight.
Prompt :	`[Mask]` News: `[Input]`
Rule :	{`[Mask]` == *Liverpool* $\vee$ *Team* $\vee$ *Football* $\vee$ *Sports*} $\wedge$ $\{s_{t,j} \geq$ threshold$\}$ $\rightarrow$ **Sports**

top-n large-error instances $\mathcal{X}_e = \{\boldsymbol{x}_{e_i}\}_{i=1}^{n}$. By proposing rules from such instances, we aim to discover novel rules that can complement the current rule set and model ensemble most effectively.

Prompt-based rule proposal. For a wide range of NLP tasks such as relation extraction and text classification, we can leverage prompts to construct informative rule templates, which naturally leads to expressive labeling rules for WSL.

Motivated by this, we design a rule proposal module based on PLM *prompting*. We present concrete examples of our prompt-based rules in Table 2.1. The input instance comes from the large-error instances identified on the clean dataset $\mathcal{D}_l$. For each task, we have a task-specific template to reshape the original input for prompting PLMs. The resulting prompt typically includes the original input as the context and a mask token to be filled by the PLMs. The final rule encompasses multiple atomic parts to capture different views of information. Each rule is accompanied by a ground-truth label of the original input instance, such a label will be assigned to the unlabeled instances matched by this rule.

For example, as shown in Table 2.1, the prompt of the relation extraction task can be "*entity* `[MASK]` *entity*", which rephrases the original input using relation phrases while

keeping the key semantics. Take news topic classification as another example, by filling the masked slot in the prompt, PLMs propose candidate keyword-based rules for topic classification. Different from the rules extracted from surface patterns of the corpus (*e.g.*, n-gram rules), such a prompt-based rule proposal can generate words that do not appear in the original inputs—this capability is important to model generalization.

Given a large-error instance $\boldsymbol{x}_{e_i} \in \mathcal{X}_e$, we first convert it into a prompt by $\boldsymbol{x}_{p_i} = \tau(\boldsymbol{x}_{e_i})$. Such a prompt consists of the key components of the original input and a `[MASK]` token. By inheriting the original input, we construct context for the `[MASK]` token to be predicted by a pre-trained LM $\mathcal{M}$. To complete the rule, we feed each $\boldsymbol{x}_{p_i}$ to $\mathcal{M}$ to obtain the probability distribution of the `[MASK]` token over the vocabulary $\mathcal{V}$:

$$p(\texttt{MASK} = \hat{\mathbf{v}} \mid \boldsymbol{x}_{p_i}) = \frac{\exp\left(\hat{\mathbf{v}} \cdot \mathcal{M}(\boldsymbol{x}_{p_i})\right)}{\sum\limits_{\mathbf{v} \in \mathcal{V}} \exp\left(\mathbf{v} \cdot \mathcal{M}(\boldsymbol{x}_{p_i})\right)}, \tag{2.4}$$

where $\mathcal{M}(\cdot)$ denotes the output vector of $\mathcal{M}$, $\mathbf{v}$ is the embedding of the token in the vocabulary $\mathcal{V}$, and $\hat{\mathbf{v}}$ is the embedding of the predicted masked token. We collect the top-k predictions with highest $p(\texttt{MASK} = \hat{\mathbf{v}} \mid \boldsymbol{x}_{p_i})$ to form the candidate rules. By filling the rules based on $\boldsymbol{x}_{e_i}$ with the prompt predictions, we obtain the candidate rule set in iteration t, denoted as $\mathcal{R}_t = \{r_j\}_{j=1}^{\mathcal{B}}$.

2.4.2 Rule Annotation and Matching

Interactive rule evaluation. As the candidate rules $\mathcal{R}_t$ can be still noisy, PRBOOST thus presents $\mathcal{R}_t$ to humans for selecting high-quality rules. Specifically, for each candidate rule $r_j \in R_t$, we present it along with its prompt template $\boldsymbol{x}_{p_j}$ to human experts, then they judge whether the rule r_j should be accepted or not. Formally, r_j is associated with a label $d_j \in \{1, 0\}$. When a rule is accepted ($d_j = 1$), it will be incorporated into the accepted rule set $\mathcal{R}^+$ for later weak label generation.

Weak Label Generation. After human evaluation, the accepted rules $\mathcal{R}_t^+$ are used to match

unlabeled instances $\mathcal{D}_u$. We design a mixed soft-matching procedure for matching rules with unlabeled instances, which combines embedding-based similarity and prompt-based vocabulary similarity. The two similarities complements each other: the embedding-based similarity captures global semantics, while the prompt-based similarity captures local features in terms of vocabulary overlapping. Given a rule $r_j \in \mathcal{R}_t^+$ and an unlabeled instance $\boldsymbol{x}_u \in \mathcal{D}_u$, we detail the computations of the two similarities below.

First, the embedding similarity is computed as the cosine similarity between the rule and instance embeddings [10]:

$$s_j^a = (\mathbf{e}_u \cdot \mathbf{e}_{r_j})/(\|\mathbf{e}_u\| \cdot \|\mathbf{e}_{r_j}\|), \tag{2.5}$$

where $\mathbf{e}_u$ is the instance embedding of $\boldsymbol{x}_u$ and $\mathbf{e}_{r_j}$ is the rule embedding of r_j, both embeddings are obtained from a PLM encoder.

Next, to compute the prompt-based similarity, we feed $\tau(\boldsymbol{x}_u)$ into the prompting model (Equation 2.4) and use the top-k candidates of the `[MASK]` position as the predicted vocabulary for instance $\boldsymbol{x}_u$. We measure the vocabulary overlapping between $\mathcal{V}_u$ and $\mathcal{V}_{r_j}$ as

$$s_j^b = \mid \mathcal{V}_u \cap \mathcal{V}_{r_j} \mid /k, \tag{2.6}$$

where $\mathcal{V}_u$ is the vocabulary of instance $\boldsymbol{x}_u$ and $\mathcal{V}_{r_j}$ is the vocabulary of rule r_j. Note that for the unlabeled instance, we have $|\mathcal{V}_u| = k$, while for the rule, we have $|\mathcal{V}_{r_j}| \leq k$ because human annotators may abstain some candidate predictions.

The final *matching score* is computed by combining the above two similarities:

$$s_j = \alpha s_j^a + (1 - \alpha) s_j^b. \tag{2.7}$$

The instance $\boldsymbol{x}_u$ is matched by the rule r_j if s_j is higher than the matching threshold σ obtained on the development set. When $\boldsymbol{x}_u$ is matched by multiple rules that provide conflicting labels, we use the one with the highest matching score to assign the weak label. If $\forall j \in 1, \cdots, k$, the matching score s_j is lower than σ, we abstain from labeling the instance $\boldsymbol{x}_u$.

2.4.3 Model Training & Ensemble

In iteration t, with the new rule-matched data $\mathcal{D}_r$, we obtain an enlarged weakly labeled dataset $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \mathcal{D}_r$. We fit a weak model m_t on $\mathcal{D}_t$ by optimizing:

$$\min_{\theta} \frac{1}{|\mathcal{D}_t|} \sum_{(\boldsymbol{x}_i, \hat{y}_i) \in \mathcal{D}_t} \ell_{\mathrm{CE}} \left(m_t(\boldsymbol{x}_i), \hat{y}_i \right), \tag{2.8}$$

where $\hat{y}_i$ is the weak label for instance $\boldsymbol{x}_i$, and ℓ_{CE} is the cross entropy loss.

While the weakly labeled dataset has been enlarged, there are still unmatched instances in D_u. To exploit such unlabeled and unmatched instances, we adopt the self-training technique for weak model training [47]. The self-training process can propagate information from the matched weak labels to the unmatched instances to improve the model m_t. Following previous models [48, 9], for each instance $\boldsymbol{x}_i \in \mathcal{D}_u$, we generate a soft pseudo-label $\widetilde{\boldsymbol{y}}_{ij}$ from the current model m_t:

$$\widetilde{\boldsymbol{y}}_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j' \in \mathcal{Y}} (q_{ij'}^2 / f_{j'})}, \quad f_j = \sum_i q_{ij} \tag{2.9}$$

where $q_i = m_t(\boldsymbol{x}_i)$ is a probability vector such that $q_i \in \mathbb{R}^K$, and q_{ij} is the j-th entry, $j \in 1, \cdots, K$.

The above process yields a pseudo-labeled $\widetilde{\mathcal{D}}_u$. We update m_t by optimizing:

$$\mathcal{L}_c(m_t, \widetilde{\boldsymbol{y}}) = \frac{1}{|\widetilde{\mathcal{D}}_u|} \sum_{\boldsymbol{x} \in \widetilde{\mathcal{D}}_u} \mathcal{D}_{\mathrm{KL}}(\widetilde{\boldsymbol{y}} \| m_t(\boldsymbol{x})), \tag{2.10}$$

where $\mathcal{D}_{\mathrm{KL}}(P \| Q) = \sum_k p_k \log(p_k / q_k)$ is the Kullback-Leibler divergence.

Finally, we incorporate the self-trained weak model into the ensemble model. The final model is a weighted ensemble of the weak models:

$$f_\theta(\cdot) = \sum_t^n \alpha_t m_t, \tag{2.11}$$

where a weak model m_t with a low error rate err_t will be assigned a higher coefficient α_t according to Equation 2.2.

Table 2.2: Main results on four benchmark datasets. †: we use different proportions of clean data for fine-tuning as described in Section subsection 2.5.1. We use gray background to show the results of WLS baselines fine-tuned on the clean data. We highlight the best fine-tuned results with purple font, and the best WSL results with blue font.

Method (Metrics)	TACRED (F1)	DBpedia (Acc.)	ChemProt (Acc.)	AG News (Acc.)
Supervised Baselines				
PLM w. 100% training data	66.9 (66.3/67.6)	99.4	79.7	94.4
PLM w. limited training data†	32.9 (40.8/27.6)	98.0	59.4	86.4
Weakly Supervised Baselines				
Rule Matching	20.1 (**85.0**/11.4)	63.2	46.9	52.3
Snorkel [14]	39.7 (39.2/40.1)	69.5	56.4	86.2
LOTClass [8]	—	91.1	—	86.4
COSINE [9]	39.5 (38.9/40.3)	73.1	59.8	87.5
Snorkel + fine-tuning†	40.8 (41.0/40.6)	97.6	64.9	87.7
LOTClass + fine-tuning†	—	98.1	—	88.0
COSINE + fine-tuning†	41.0 (40.4/41.7)	97.9	65.7	88.0
PRBOOST	**48.1** (42.7/**55.1**)	**98.3**	**67.1**	**88.9**

2.5 Experiments

2.5.1 Experiment Setup

Tasks and Datasets We conduct experiments on four benchmark datasets, including *TACRED* [49] for relation extraction, *DBPedia* [50] for ontology classification, *ChemProt* [51] for chemical-protein interaction classification and *AG News* [50] for news topic classification. For the initial weak supervision sources, we use the labeling rules provided by existing works: [10] for TACRED, [8] for DBPedia, and [20] for Chemprot and AG News. The statistics of the four datasets are shown in Table C.1. For the development set, we do not directly use the full development set as suggested by the recent works [41, 52]. This prevents the model from taking the advantage of the massive number of labeled data in the development set. Instead, we create a real label-scarce scenario and keep the number of sample in validation set $\mathcal{D}_v$ the same as the limited clean labeled set $\mathcal{D}_l$, namely $|\mathcal{D}_v| = |\mathcal{D}_l|$.

Baselines. We include three groups of baselines:

Fully Supervised Baseline: **PLM**: We use the pretrained language model RoBERTa-base [53] as the backbone and fine-tune it with the full clean labeled data except for

ChemProt. On ChemProt, we choose BioBERT [54] as the backbone for all the baselines and our model to better adapt to this domain-specific task. The performance of fully supervised methods serves as an upper bound for weakly-supervised methods.

Weakly Supervised Baselines: (1) **Snorkel** [14] is a classic WSL model. It aggregates different labeling functions with probabilistic models, then fed the aggregated labels to PLM for the target task. (2) **LOTClass** [8] is a recent model for weakly-supervised text classification. It uses label names to probe PLMs to generate weak labels, and performs self-training using the weak labels for classification. (3) **COSINE** [9] is a state-of-the-art method on fine-tuning PLMs with weak supervision. It adopts self-training and contrastive learning to fine-tune LMs with weakly-labeled data.

Interactive Learning Baselines: (1) **Entropy**-based AL [23] is a simple-yet-effective method for AL which acquires samples with the highest predictive entropy. (2) **CAL** [27] is the most recent method for active learning. It selects samples has the most diverge predictions from their neighbors for annotation. (3) **IWS** [18] is an interactive WSL model. It firstly generates n-gram terms as candidate rules, then selects quality rules by learning from humans' feedback. Note that IWS is designed for binary classification, which makes it hard to adapt to classification with multiple labels.

Evaluation Protocol. To propose rules on large-error instances, we assume access to a dataset $\mathcal{D}_l$ with a limited number of clean labeled data. For our method, such a clean dataset is only used for identifying large-error instances. For fair comparison, for the WSL baselines, we further fine-tune them using the same clean data and compare with such fine-tuned results. Specifically, we use 5% clean data for TACRED and ChemProt, 0.5% for AG News and 0.1% for DBPedia. We then implement a 10-iteration rule proposal and weak model training. In each iteration, we identify the top-10 large-error instances and propose 100 candidate rules in total (*i.e.*, 10 candidate rules per instance). Each rule is annotated by three humans, and the annotated rule labels are majority-voted for later weak label generation. Following the common practice [49, 20], we use F1 score for TACRED and accuracy for other datasets.

2.5.2 Main Results

Table 2.2 shows the performance of PRBOOST and the baselines on the four datasets. The results show that PRBOOST outperforms the weakly supervised baselines on all the four datasets. When the weakly supervised baselines are not fine-tuned on $\mathcal{D}_l$, PRBOOST outperforms the strongest WSL baseline by 8.4%, 7.2%, 7.3%, 2.4% on the four benchmarks. Even when the WSL models are further fine-tuned using clean labeled data, PRBOOST still outperform them by 2.4% on average. Compared against supervised baselines, PRBOOST is significantly better than the fine-tuned model on TACRED, ChemProt and AG News when the training data is limited. For the model fine-tuned with 100% training data, we narrow the gap to fully supervised learning, compared to other WS approaches.

Comparing the performance gains across datasets, the performance gap between PRBOOST and the baselines is the largest on TACRED, which is the most challenging task among the four with 41 different relation types. ChemProt is the smallest dataset with only 5400 training data, so the gain is larger when the WSL methods are fine-tuned with clean labels. The performance gaps among different methods are small on DBPedia, especially after they are fine-tuned using clean labeled data. DBpedia, being a relatively simple dataset, using only 0.1% clean data for fine-tuning RoBERTa already achieves 98% accuracy, and the other WSL methods after fine-tuning perform similarly.

It is worth noting that PRBOOST performs strongly across all the tasks because we can easily design a task-specific prompt template to adapt to each task. In contrast, some WSL baselines are difficult to apply to certain tasks. For example, LOTClass achieves strong performance for DBpedia and AGNews as its weak sources are tailored for text classification. However, it is hard to apply it to relation extraction tasks. Similarly, IWS performs well on binary classification problems using n-gram based rules, but the method is only designed for binary classification, making it unsuitable for complex multi-class tasks.

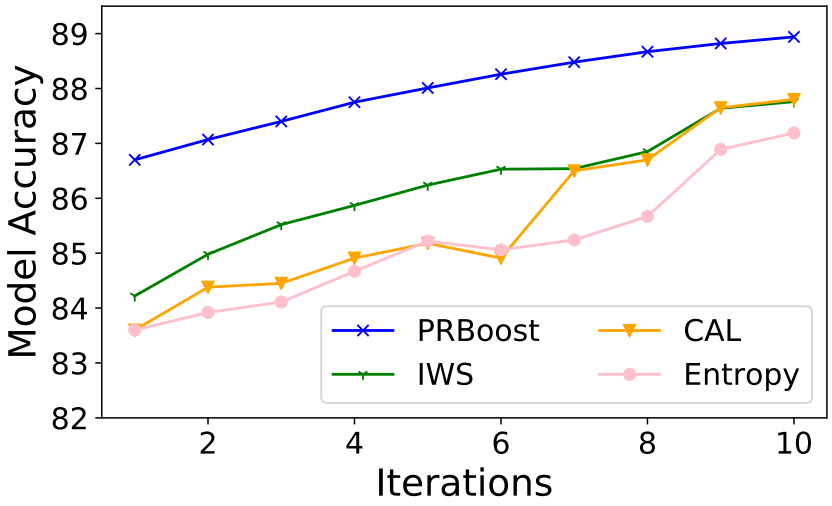


Figure 2.2: Comparison of PRBOOST and interactive methods on AG News.

2.5.3 Rule Annotation Agreement and Cost

In this set of experiments, we benchmark model performance and annotation cost against interactive learning baselines (detailed in section B.1): IWS, CAL, and Entropy-based AL. As shown in Figure 2.2, PRBOOST outperforms IWS that also features rule-level annotation by 1.2% with very close annotation cost. Our method outperforms the best interactive baseline CAL by 1.1% in terms of accuracy, while using about $0.6\times$ annotation cost. While annotating model-proposed rule or instances, we asked all the three annotators to time their annotation. On average, it takes each annotator less than 3 seconds to annotate one rule, while it takes nearly 10 seconds to annotate one instance. Rule-level annotation is much more efficient than instance-level annotation because 1) we show the prompt rather than the original instance to humans, which is shorter and easier to read; 2) upon scanning the prompt, the annotators can swiftly select qualified rules as they only differ at the `[MASK]` position. This shows that rule-level annotation is an efficient and suitable paradigm for interactive WSL.

For the annotation agreement, we compute Fleiss' kappa κ [55] to evaluate the agreement among multiple human annotators. This statistic assesses the reliability of agreement among

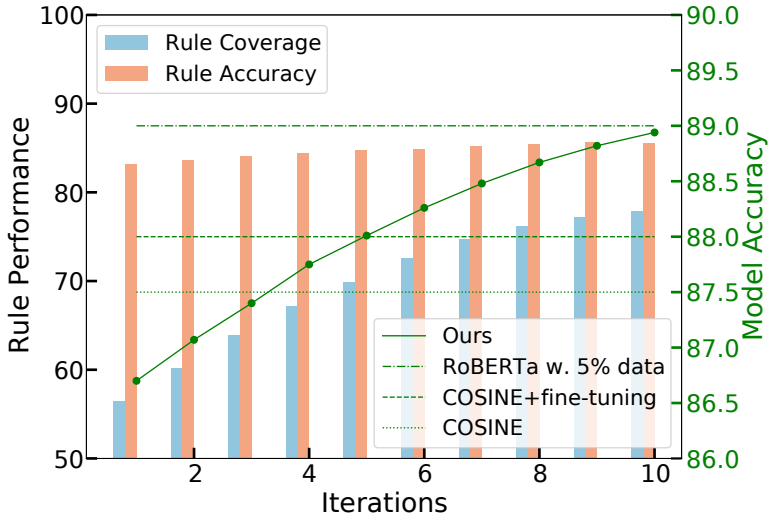


Figure 2.3: Rule performance and model accuracy vs. iterations on AG News.

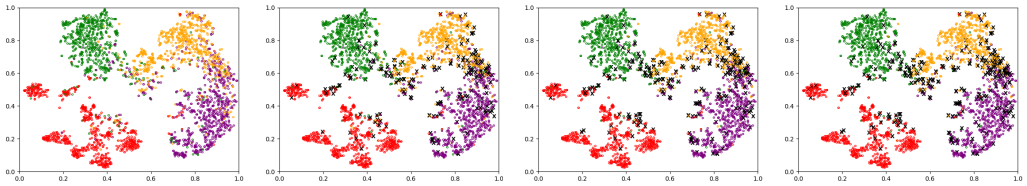
Table 2.3: Annotation agreement measured by the Fleiss-Kappa κ on AG News. $\bar{P}$ measures annotation agreement over all categories; $\bar{P}_e$ computes the quadratic sum of the proportion of assignments to each category.

Iteration	1	2	3	4	5	6	7	8	9	10	Overall
$\bar{P}$	.89	.90	.93	.90	.87	.92	.91	.91	.87	.90	.90
$\bar{P}_e$	.63	.59	.73	.71	.62	.73	.66	.56	.68	.68	.65
κ	.71	.77	.73	.66	.65	.71	.75	.79	.60	.68	.71

multiple annotators. $\kappa = 1$ indicates complete agreement over all the annotators, and no agreement results in $\kappa \leq 0$. As shown in Table 2.3, we obtained an average $\kappa = 0.71$, which means the annotators achieve substantial agreement. For each iteration, the κ ranges between $[0.60, 0.79]$ indicating the stability of the annotation agreement.

2.5.4 Rule Quality in Iterative Learning

In this set of experiments, we evaluate the quality of the rules discovered by PRBOOST. Figure 2.4 visualizes the discovered rules on AG News dataset. We observe that 1) the rules can rectify some mis-classified data, and 2) the rules can complement each other. For the first observation, we can take Figure 2.4a and Figure 2.4b for example. In iteration 0 where new rules have not been proposed, it is obvious that some green data points and purple data points are mixed into the orange cluster. After the first-round rule proposal, PRBOOST has



(a) Iteration 0 (b) Iteration 1 (c) Iteration 4 (d) Iteration 10

Figure 2.4: T-SNE visualization [56] of rule-matched data that are mis-classified by the model on AG News dataset. The four classes are represented by different colors, and the black cross denotes the rule-matched data.

already rectified parts of wrong predictions via rule-matching. This is because our rule proposal is targeted on the large-error instances, such adaptively discovered rules can capture the model's weakness more accurately compared to the simply enumerated rules. For the second observation, we found that more mis-classified data points get matched by the newly discovered rules as the iteration increases. It demonstrates PRBOOST can gradually enlarge the effective rule set by adding complementary rules, which avoids proposing repetitive rules that can not improve the rule coverage.

Figure 2.3 shows the changes in rule accuracy, rule coverage, and model performance in the iterative learning process on AG News. As shown, the model's accuracy increases steadily during learning, which is improved from 86.7% to 88.9% after 10 iterations. This improvement arises from two key aspects of PRBOOST. First, the enlarged rule set continuously augments weakly labeled data, which provides more supervision for the weak model training. Second, the model ensemble approach refines the previous large errors step by step, resulting in increasing ensemble performance.

Regarding the rule coverage and accuracy, we observe the coverage of the rule set is improved from 56.4% to 77.8%, and rule accuracy from 83.1% to 85.6%. Such improvements show that PRBOOST can adaptively propose novel rules to complement the previous rule set, which can match more instances that were previously unmatchable. Note that the increased rule converge has not compromised rule accuracy, but rather improved it. The reason is twofold: (1) the human-in-the-loop evaluation can select high-quality rules for generating new

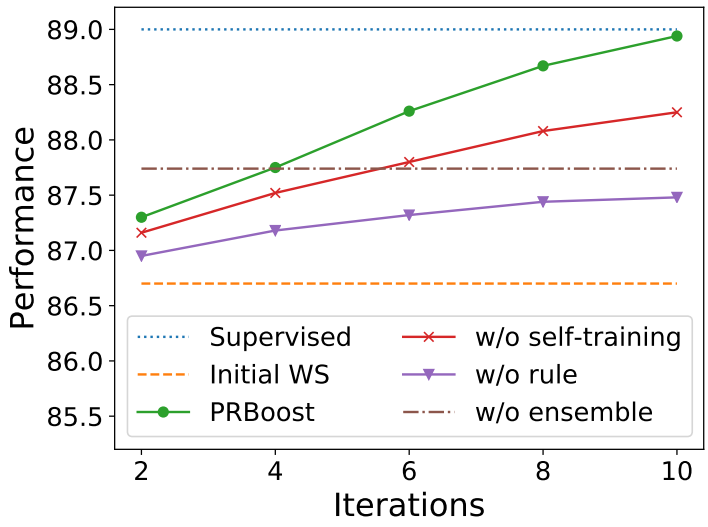


Figure 2.5: Ablation study on AG News. The three horizontal lines represent the no-iterative methods. We use COSINE as the initial WS baseline. For the supervised baseline, we fine-tune RoBERTa on 5% clean data.

weak labels; (2) for the instances with wrong initial weak labels, PRBOOST can discover more rules for the same instances and correct the weak labels through majority voting.

2.5.5 Ablation Study

We study the effectiveness of various components in PRBOOST and show the ablation study results in Figure 2.5. We have the following findings:

First, the boosting-based iterative rule discovery strategy is effective. For the "w/o ensemble" setting, we fix the annotation budget $\mathcal{B}$ but discover candidate rules from large-error samples in one iteration. The results show the superiority of the iterative strategy in PRBOOST , which brings 1.2% performance gain. PRBOOST iteratively identifies the current model's weaknesses and proposes rules to strengthen itself, therefore it adaptively discovers more effective rules than static rule discovery.

Second, ensembling alone without new rule discovery is not as effective. For the "w/o rule" variant, we do not propose new rules, but ensemble multiple self-trained weak classifiers instead. The final performance drops significantly under this setting by 1.5%. It

demonstrates the newly proposed rules provide complementary weak supervision to the model. Although simply ensembling multiple weak classifiers also helps WSL, it is not as effective as training multiple complementary weak models as in PRBOOST.

Third, self-training benefits learning from new weak labels. For the "w/o self-training" setting, we do not use the self-training technique when learning each weak classifier. The performance deteriorates by 0.6%. This is because part of the data are still unmatched after we propose new rules, and self-training leverages the unlabeled data to help the model generalize better.

2.6 Conclusion

We proposed PRBOOST to iteratively discover prompt-based rules for interactive weakly-supervised learning. Through a boosting-style ensemble strategy, it iteratively evaluates model weakness to identify large-error instances for new rule proposal. From such large-error instances, its prompt-based rule discovery module leads to expressive rules that can largely improve rule coverage while being easy to annotate. The discovered rules complement the current rule set and refine the WSL model continuously. Our experiments on four benchmarks demonstrate that PRBOOST can largely improve WSL and narrow the gaps between WSL models and fully-supervised models.

CHAPTER 3
IMPROVING DATA UTILIZATION EFFICIENCY VIA DATA WEIGHTS OPTIMIZATION

3.1 Introduction

Aligning large language models (LLMs) with human preferences is crucial to ensure LLMs adhere to human values. Recent advances in LLM alignment have primarily focused on algorithmic innovations, ranging from reinforcement learning from human feedback (RLHF) [57] to alternative preference learning approaches like sequence likelihood calibration (SLiC) [58] and direct preference optimization (DPO) [59]. Although such algorithmic advancements have enhanced LLM alignment, the quality of preference data used for reward modeling has received limited attention.

Recent studies highlight two key challenges in reward modeling for LLM alignment: (1) **Preferene data are often noisy**: [60] report 19-37% preferences provided by crowd workers are noisy. [5] reveal that preference noise is observed in a wide range of tasks, including QA, Summarization, and Dialogue, where the noise rate ranges between 20-40%. (2) **Not all preference data contributes equally**: Noisy preference data forces reward models to require more training data to achieve desired performance. For instance, [61] categorize preference data into groups based on preference strength and find that some groups negatively impact reward modeling, which highlights the need for selective data curation to boost reward model performance.

Existing works addressing preference data quality fall into two categories, each with significant limitations. The first category, *heuristic-based filtering*, employs predefined quality criteria to remove low-quality data. For example, [62] retains only top-scored data points based on reward model predictions, potentially discarding informative examples that

could aid in learning decision boundaries. [63] develop dataset-specific filtering rules based on response length, semantic similarity, and sentiment analysis. While effective, these rigid, manually crafted criteria often fail to capture the nuanced aspects of preference quality and require substantial human effort to adapt to new domains. On the other hand, *denoising techniques* attempts to mitigate the impact of noisy samples during model training. [64] leverage LLMs to self-refine the reward difference between positive and negative pairs based on the data quality assessed through the LLMs-as-judges framework. [65] compute the KL divergence between predicted preferences and annotated labels to retain reliable preference labels and flip unreliable ones. However, these methods struggle in achieving both robust denoising performance and high data efficiency.

These existing approaches face three key limitations: (1) they often rely on heuristics or expensive oracle feedback, (2) they treat ambiguous preferences as unreliable samples rather than potential learning opportunities, and (3) they fail to adaptively adjust data importance based on the reward modeling progress. These limitations raise a critical research question: Can we develop a learning-based approach to automatically estimate the importance of preference data, thereby enhancing reward modeling?

We address this challenge by introducing DORM (preference **D**ata weights **O**ptimization for **R**eward **M**odeling) – a two-stage approach that combines preference data quality estimation with adaptive data weighting. The key intuition of the first stage is that data points where the model is uncertain are often informative and should be emphasized, whereas mislabeled or unreliable samples should be down-weighted. To this end, we estimate preference data weights by integrating *model uncertainty* and *prediction disagreement*. Specifically, we employ approximate Bayesian inference techniques for uncertainty estimation [66], while measuring disagreement by the discrepancy between predictions and labels. This procedure prioritizes informative and reliable data and mitigates the impact of unreliable data.

The second stage further refines data weighting by dynamically adjusting weights through a bilevel optimization framework. This approach maximizes validation performance

while using initial uncertainty-based data weights as regularization. The framework consists of two levels: (1) The *upper-level optimization* adjusts data weights to minimize validation loss, guided by initial weight estimates; (2) The *lower-level optimization* trains a reward model using the weights determined by the upper-level problem. This learning process automatically identifies which data points are most useful for improving validation performance while mitigating overfitting through uncertainty-based regularization.

To evaluate our method, we conduct experiments on both the reward model and policy model levels. Our 12B reward model, trained with only 50k preference data samples, achieves 90.2% overall performance on RewardBench, matching the performance of similar-sized models trained with significantly more data. Analysis of data weight tracking reveals that high-quality datasets are progressively assigned greater weights, while noisy samples are gradually down-weighted. Furthermore, the policy model aligned using our reward model shows 61.2% win rates compared to using the baseline reward model.

Our contributions can be summarized as follows: 1) **Preference Data Quality Estimation:** We introduce a method for estimating preference data quality by combining epistemic uncertainty and disagreement measures, providing an uncertainty-based assessment of data importance. 2) **Bilevel Optimization Framework:** We formulate learning to weigh preference data as a bilevel optimization problem, enabling data-driven refinement of data weights while incorporating initial quality estimates. 3) **Data-Efficient Reward Model Training:** Our approach achieves 90.5% accuracy on RewardBench, with $10 - 40\times$ less preference data compared to baselines.

3.2 Related Work

3.2.1 Preference Learning for LLM Alignment

Preference learning is a fundamental component of aligning LLMs with human intent, enabling models to rank and prioritize responses based on human feedback. Reinforcement Learning from Human Feedback (RLHF) [57, 67, 68] is the most widely used approach,

where a reward model is trained on human preferences and used to guide policy optimization. However, RLHF is prone to issues such as reward hacking [69, 70] and unstable training [71, 72].

To address the limitations of RLHF, alternative preference learning methods have been proposed. Direct Preference Optimization (DPO) [59] directly optimizes policy models based on human preferences without requiring an explicit reward model, reducing reward over-optimization. Sequence Likelihood Calibration (SLiC) [58] refines preference learning by leveraging likelihood-based objectives to better align model outputs. Additionally, methods such as RLAIF [73] and reward model distillation [74] propose refining alignment by integrating preference signals without the instability associated with reinforcement learning.

A few recent works have explored incorporating uncertainty estimation into reward models to improve alignment reliability. The Uncertainty-aware Reward Model (URM) [75] introduces a probabilistic value head to model aleatoric uncertainty and leverages an ensemble-based approach (URME) to quantify epistemic uncertainty by examining discrepancies among individual reward models. However, URME requires multiple reward models, making it costly for deployment. Differently, our method performs uncertainty estimation with a single reward model, ensuring deployment efficiency. Additionally, our approach is compatible with various uncertainty estimation methods and extends beyond uncertainty-based quality estimation by incorporating dynamic data weighting throughout reward modeling.

The above methods primarily focus on learning from available preference data rather than addressing its quality. In contrast, our work introduces a quality-aware preference weighting mechanism that dynamically adjusts data importance, enhancing reward modeling without reliance on external heuristics or oracle feedback.

3.2.2 Quality-aware LLM Alignment

Recent studies have highlighted the impact of preference data quality on alignment performance [76, 77, 78]. [5] demonstrated that a significant portion of preference pairs in LLM alignment datasets contain noise, affecting alignment outcomes. [61] emphasized the importance of identifying and prioritizing informative preference data for effective alignment.

To address these data quality issues, heuristic-based filtering methods aim to select high-quality data based on predefined creteria. [62] deploy a reward model to score preference data samples and keep only the top-scored ones, potentially discarding informative examples that could aid in learning decision boundaries. [63] employ dataset-specific rules based on lexical or semantic patterns to filter low-quality data from preference collections. While effective to some extent, these approaches often require considerable human effort and may not generalize well across diverse datasets.

Denoising techniques have also been explored to improve data quality. [64] leverage LLMs to self-refine reward differences between positive and negative pairs. [65] introduce a denoising discriminator for robust reward learning, where the discriminator filters preference labels by using a dynamic lower bound on the KL divergence between predicted preferences and annotated labels to retain trustworthy samples, while applying an upper bound to flip unreliable labels. These methods aim to improve data quality during the training process but face challenges in balancing sample efficiency and robustness.

An alternative direction is synthetic data generation. Methods proposed by [79] and [80] replace human feedback with AI-generated feedback conditioned on human-written principles. While this can potentially scale the creation of preference data, it introduces new challenges in ensuring the quality and diversity of synthetic data.

Despite their advantages, existing approaches remain constrained by key limitations as discussed in section 3.1: they often depend on heuristic rules or external oracle feedback, struggle to leverage ambiguous preferences as informative training signals, or lack

mechanisms to dynamically adjust data importance throughout training. To overcome these challenges, DORM equips a quality-aware preference data weighting strategy and enables dynamic weighting in the reward modeling process.

3.3 Preliminaries

Multi-Objective Reward Modeling. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the space of prompts and responses, respectively. Unlike traditional reward models that rely solely on pairwise preferences [81, 57, 74, 59], multi-objective reward modeling leverages fine-grained ratings across multiple reward attributes [82, 83, 84] to capture richer preference signals. In this setup, human feedback is collected in the form of structured ratings, where each response $y \in \mathcal{Y}$ is assigned a rating vector $r \in \mathbb{R}^m$, with each dimension corresponding to a different reward attribute.

Given a dataset $\mathcal{D} = \{(x_i, y_i, \mathbf{r_i})\}_{i=1}^{|\mathcal{D}|}$, where x_i represents the input prompt, y_i is the generated response, and $\mathbf{r_i}$ is the associated multi-dimensional rating, the objective is to learn a reward model r_θ that predicts these ratings accurately. Specifically, given an input pair (x, y), we concatenate the prompt and response as $x \oplus y$ and pass it through a pre-trained decoder-only LLM $\mathcal{M}_\psi$, extracting a d-dimensional representation from the final decoder layer. A linear regression head $V \in \mathbb{R}^{d \times m}$ is then applied to produce the predicted rating vector, which is optimized using the following regression loss:

$$\min_{\psi, V} \mathbb{E}_{x,y,\mathbf{r}\sim\mathcal{D}} \left\| V^\top f_\psi(x \oplus y) - \mathbf{r} \right\|_2^2. \tag{3.1}$$

Projection for Scalar Metrics. Since the reward model operates in a multi-objective setting, where each response y is evaluated across m different reward attributes, it is often necessary to project the reward vector $\mathbf{r} \in \mathbb{R}^m$ onto a scalar space. To achieve this, we introduce a projection vector $\lambda \in \mathbb{R}^m$, which allows us to map the multi-dimensional reward vector onto a single scalar value by $r = \lambda^\top \mathbf{r}$.

Quality-based Data Weighting. Given the dataset $\mathcal{D}$, we aim to estimate the quality of each data point to derive appropriate weights for reward modeling. Let $s_i \in \mathbb{R}$ denote the estimated quality score for each sample $(x_i, y_i, \mathbf{r_i})$. The corresponding data weight w_i is derived as $w_i = h(s_i)$, where $h : \mathbb{R} \to \mathbb{R}^+$ is a monotonically increasing function that maps quality measures to data weights. These weights are then incorporated into the reward modeling objective:

$$\min_{\psi, V} \mathbb{E}_{(x,y,\mathbf{r})\sim\mathcal{D}} w \left\| V^\top f_\psi(x \oplus y) - \mathbf{r} \right\|_2^2. \tag{3.2}$$

This formulation allows the model to adaptively focus on samples based on their estimated quality during training.

Direct Preference Optimization. To align a policy model π_ϕ with human preferences, RLHF [57] employs a reward model to score generated responses, followed by policy optimization using Proximal Policy Optimization (PPO) [85]. However, PPO-based training is often unstable [71, 72], requires extensive hyperparameter tuning, and demands significant computational resources.

To mitigate these challenges, DPO [59] provides a more efficient alternative. For a preference dataset $\mathcal{D} = \{(x_i, y_i^w, y_i^l)\}_{i=1}^{|\mathcal{D}|}$, where each triplet consists of a prompt x_i, a preferred response y_i^w, and a less preferred response y_i^l, DPO defines an implicit reward function derived from the log-probabilities of the policy π_ϕ relative to a reference policy π_{ref}. The optimization objective is:

$$\begin{aligned} &\mathcal{L}_{\text{DPO}}\left(\pi_\phi; \pi_{\text{ref}}\right) = -\mathbb{E}_{\left(x, y^w, y^l\right)\sim\mathcal{D}} \\ &\left[\log \sigma\left(\beta \log \frac{\pi_\phi\left(y^w \mid x\right)}{\pi_{\text{ref}}\left(y^w \mid x\right)} - \beta \log \frac{\pi_\phi\left(y^l \mid x\right)}{\pi_{\text{ref}}\left(y^l \mid x\right)}\right)\right], \end{aligned} \tag{3.3}$$

where $\sigma(\cdot)$ denotes the sigmoid function, and β is a temperature parameter controlling the sharpness of preference modeling.

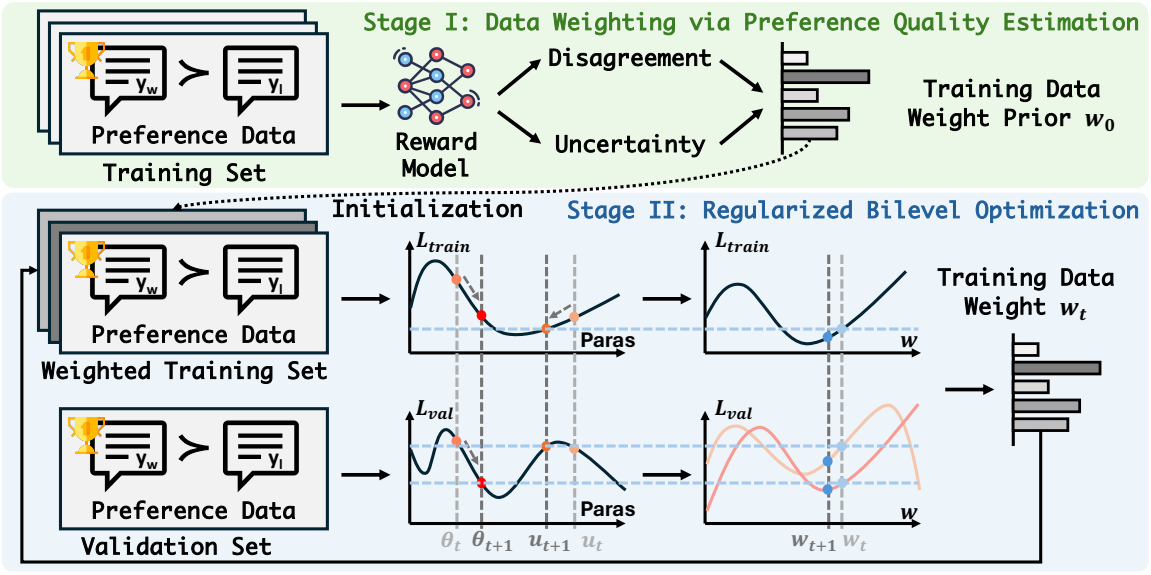


Figure 3.1: Overview of the two-stage bilevel optimization framework for reward modeling. The first stage assigns initial data weights based on epistemic uncertainty and disagreement. The second stage refines these weights in a bilevel optimization process, using a validation set and the initial weights as regularization.

3.4 Preference Data Weight Optimization with Estimated Quality

In this section, we formulate the weight optimization for preference data as a bilevel optimization problem. Our approach operates in two stages. In the first stage, we estimate initial data weights based on epistemic uncertainty [66, 86] and a disagreement function. These measures help identify more reliable and informative data points, guiding the model towards higher-quality examples. The second stage employs a bilevel optimization framework [87]. The lower level optimizes reward model parameters using weighted preference data, while the upper level refines these weights to minimize loss on a high-quality validation set. Critically, we incorporate the initial weights as regularization terms in the upper-level problem, anchoring the optimization to the prior quality estimates and stabilizing the process. We present our method overview in Figure 3.1.

3.4.1 Quality-aware Preference Data Weighting

In the context of preference data optimization, it is crucial to account for the varying quality of data samples [61]. Assigning appropriate weights to each data point based on its estimated quality can enhance model performance by emphasizing informative samples and down-weighting noisy or unreliable ones. To achieve this, we propose a method to estimate prior weights for each data point by leveraging measures of *epistemic uncertainty* and *prediction disagreement*.

Estimating Epistemic Uncertainty. For each data point x_i, we estimate the epistemic uncertainty using Monte Carlo (MC) dropout [66][1]. We perform N stochastic forward passes through the reward model, where dropout introduces randomness by deactivating different subsets of neurons in each pass. This process yields a set of predictions $\{\hat{r}_{i,1}, \hat{r}_{i,2}, \ldots, \hat{r}_{i,N}\}$ for the data point x_i.

To compute the uncertainty, we first compute the mean prediction by $\bar{r}_i = \frac{1}{N}\sum_{n=1}^{N} \hat{r}_{i,n}$, and then take the variance as uncertainty:

$$u_i = \sigma_i^2 = \frac{1}{N}\sum_{n=1}^{N} \left(\hat{r}_{i,n} - \bar{r}_i\right)^2 . \tag{3.4}$$

Here, u_i represents the epistemic uncertainty associated with x_i. A higher variance u_i indicates that the model's predictions are more sensitive to changes in its internal parameters, reflecting less confidence in its output for that data point.

Measuring Prediction Disagreement. To assess the alignment between the model's predictions and labels, we define a *disagreement function* q_i for each data point as:

$$q_i = \frac{|r_i - \bar{r}_i|}{\max_n \hat{r}_{i,n} - \min_n \hat{r}_{i,n}}, \tag{3.5}$$

[1]While we use MC dropout for simplicity and computational efficiency, our method can work with other Bayesian uncertainty estimation techniques.

where r_i is the label, $\bar{r}_i$ is the mean prediction from the MC dropout ensemble, the normalization ensures that q_i is scale-invariant.

A higher value of q_i suggests a greater discrepancy between the model's prediction and the label, potentially indicating mislabeling or noise in the data.

Defining Prior Weights. To effectively prioritize data points during training, we combine the epistemic uncertainty u_i and the disagreement measure q_i to formulate the prior weight w_i^0 for each data point:

$$w_i^0 = \exp\left(u_i - \gamma q_i\right), \tag{3.6}$$

where $\gamma > 0$ is a hyperparameter controlling the balance of uncertainty and disagreement.

This formulation is grounded in the intuition that data points with high epistemic uncertainty (large u_i) are situated in regions where the model lacks confidence but have the potential to provide significant learning gains. By assigning higher weights to these points, we encourage the model to focus on inputs where additional information could most improve its performance.

Conversely, the disagreement measure q_i serves as a penalty term. Data points where the model's predictions significantly deviate from the labels (large q_i) may indicate mislabeling or noisy data. By subtracting γq_i in the exponent, we reduce the weights of these potentially unreliable points, thereby mitigating their influence on the training process.

Interpretation in Extreme Cases. The prior weight formulation behaves intuitively under extreme scenarios. When $u_i \to \infty$ and $q_i \approx 0$, the weight w_i^0 becomes very large. This indicates that the data point is highly informative (high uncertainty) and reliable (low disagreement), deserving significant emphasis during training. Conversely, when $u_i \approx 0$ and $q_i \to \infty$, the weight w_i^0 approaches zero. In this case, the model is confident in its prediction (low uncertainty) but disagrees with the label (high disagreement), suggesting possible mislabeling; down-weighting such points prevents them from negatively impacting

the model. For data points where both uncertainty and disagreement are high, the weight depends on the relative values of u_i and γq_i, allowing for careful consideration based on the specified hyperparameters.

Algorithm 1 Solve the Bilevel Optimization Problem via A First-order Hypergradient Method

Require: initialization w_0, u_0, θ_0, learning rates $\{\eta_w, \eta_u, \eta_\theta\}$, and coefficients α, β

1: **for** $t = 0$ **to** $T - 1$ **do**
2: Sample mini-batches $\{D_{\text{tr}}^t, D_{\text{val}}^t\}$ from the training set and the validation set $\{D_{\text{tr}}, D_{\text{val}}\}$
3: $u_{t+1} = u_t - \alpha\eta_u \nabla_u L_t r(w_t, u_t; D_{\text{tr}}^t)$
4: $\theta_{t+1} = \theta_t - \eta_\theta(\nabla_\theta L_v al(w_t, \theta_t; D_{\text{val}}^t) + \alpha \nabla_\theta L_t r(w_t, \theta_t; D_{\text{tr}}^t))$
5: $w_{t+1} = w_t - \eta_w(\nabla_w L_v al(w_t, \theta_t; D_{\text{val}}^t) + 2\beta(w_t - w_i^0) + \alpha(\nabla_w L_t r(w_t, \theta_t; D_{\text{tr}}^t) - \nabla_w L_t r(w_t, u_t; D_{\text{tr}}^t)))$
6: **end for**
7: **return** (w_T, θ_T, u_T)

3.4.2 Regularized Bilevel Optimization

To incorporate prior knowledge about data quality into our optimization framework, we include a regularization term for the upper-level optimization problem that penalizes deviations from the estimated prior weights. Specifically, we use the prior weights w_i^0 derived from the measures of epistemic uncertainty and prediction disagreement discussed earlier.

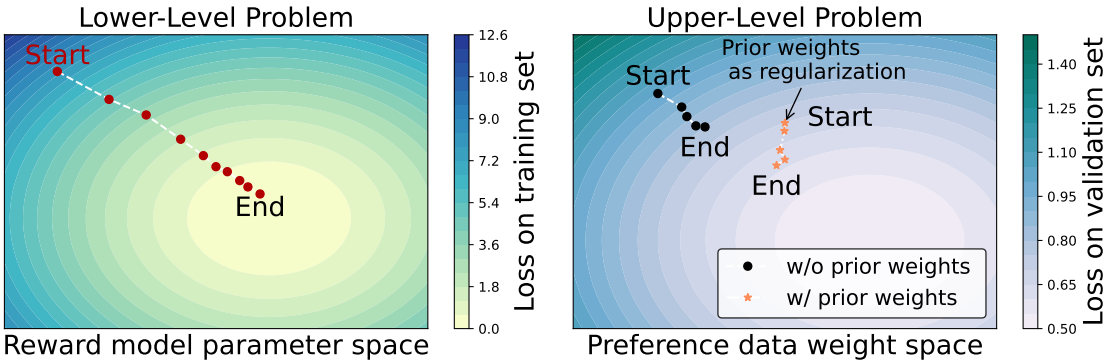


Figure 3.2: Regularized bilevel optimization for reward modeling, where the prior weights are initialized from preference data quality estimation.

Let θ represent the model parameters, and $w = [w_1, w_2, \ldots, w_k] \in \mathbb{R}^k$ be the vector of

weights assigned to each training sample s_i, where $i = 1, \ldots, k$ indexes over the training data, and k is the total number of samples. We formulate the bilevel optimization problem as:

$$\begin{aligned} &\min_{w \in \mathcal{W}} L_{val}(\theta^*(w)) + \beta \sum_{i=1}^{k} (w_i - w_i^0)^2 \\ &\text{s.t.} \quad \theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{k} w_i L_{tr}(\theta, s_i) \end{aligned} \tag{3.7}$$

where $L_{\text{tr}}(\theta, s_i)$ and $L_{\text{val}}(\theta^*(w))$ are the training and validation loss, respectively, and $\beta > 0$ is a hyperparameter controlling the strength of the regularization. This L2 regularization encourages the optimized weights to stay close to the prior weights while allowing for data-driven adjustments. The gradient with respect to each weight p_i is simply:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L_{val}}{\partial w_i} + 2\beta(w_i - w_i^0) \tag{3.8}$$

Here we choose L2 regularization instead of KL divergence because this formulation provides a simpler, more computationally efficient way to incorporate prior knowledge, avoiding the numerical instabilities associated with KL divergence while allowing for unconstrained optimization of weights.

3.4.3 Solve the Bilevel Optimization via A First-order Hypergradient Method

Solving the bilevel optimization problem in Eq. (Equation 3.7) directly is challenging due to the nested optimization and the dependency of the upper-level objective on the lower-level solution $\theta^*(w)$. To address this, we adopt a fully first-order method inspired by [87, 88], which circumvents the computational burdens associated with second-order derivatives and is well-suited for stochastic optimization settings.

We begin by reformulating the bilevel problem as a single-level optimization with an equality constraint representing the optimality of the lower-level problem:

$$\begin{aligned} &\min_{w \in \mathcal{W}, \theta} L_{val}(w, \theta) + \beta \sum_{i=1}^{k} (w_i - w_i^0)^2 \\ &\text{s.t.} \quad L_{tr}(w, \theta) - \min_{u} L_{tr}(w, u) = 0. \end{aligned} \tag{3.9}$$

Here an auxiliary variable u is introduced to transform the lower problem $\theta^*(w) = \arg\min_\theta L_t r(w, \theta)$ to be the constraint $L_t r(w, \theta) - \min_u L_t r(w, u) = 0$ where u serves as the proxy of $\theta^*(w)$. This leads to a minimax problem defined as $\min_{w \in \mathcal{W}} \max_u \mathcal{L}^\alpha(w, \theta, u)$, where

$$\begin{aligned} \mathcal{L}^\alpha(w, \theta, u) = L_{val}(w, \theta) + \beta \sum_{i=1}^{k} (w_i - w_i^0)^2 \\ + \alpha(L_{tr}(w, \theta) - L_{tr}(w, u)). \end{aligned} \tag{3.10}$$

This reformulation circumvents the upper-lower dependency in the original bilevel optimization via an equivalent min-max problem. We summarize our algorithm in Algorithm 1.

3.5 Experiments

3.5.1 Experiment Setup

Datasets. We experiment with five datasets, with details provided in section I.1.

- *HelpSteer2* [89]: the most recent high-quality human-annotated preference data, a follow-up to the popular HelpSteer.
- *OpenAssistant2* [90]: human-annotated assistant-style conversation corpus with a multi-dimension label.
- *Magpie-QWen-2.5* [91]: a synthetically generated dataset for supervised fine-tuning using Qwen2-72B-Instruct.
- *OffsetBias* [92]: a pairwise preference dataset intended to reduce common biases inherent in judge models.
- *WildGuard* [93]: with examples designed to evaluate the safety of LLM responses under various conditions.

Baselines. We consider two baseline groups:

- *LLM-as-a-judge*: These models generate a preference label given a prompt with a pair of responses as input. We include Llama-3.1-405B [67], GPT-4 [94], GPT-4o [95], Gemini-

1.5-pro [68] and self-taught evaluator [96] which is based on Llama-3-70B.

- *Standard Reward Models*: This category consists of models those explicitly trained on preference data. We compare against standard RM [97], Cohere-0514, Llama3-70B-SteerLM-RM [84], URM-Llama3-8B [75], ArmoRM-Llama3-8B-v0.1 [98], Pair-preference-model-LLaMA3-8B [63], and Internlm2-20B-reward [99].

Evaluation. We evaluate DORM at both the reward model and policy model levels. For reward model performance, we benchmark on RewardBench [100]. For policy model evaluation, we conduct experiments on UltraFeedback [101].

Implementation. We use Mistral NeMo 12B-Instruct [102] as the backbone of our reward model. We set multi-attribute head (total of 12 attributes for our recipe) with regression loss for reward modeling. Additional implementation details are provided in section I.3.

Table 3.1: Main results on RewardBench. For baseline methods, we choose those without data contamination (some models on RewardBench were reported that their training data may overlap with prompts present in the RewardBench evaluation set). † indicates models trained with at least x10 preference data.

Models	**Chat**	**Chat_Hard**	**Reasoning**	**Safety**	**Overall**
LLM-as-a-judge					
Llama3.1-405B-Instruct [67]	97.2	74.6	77.6	87.1	84.1
GPT-4-0125 [94]	95.3	74.3	87.6	86.9	86.0
GPT-4o-0806 [95]	96.1	76.1	88.1	86.6	86.7
Gemini-1.5-pro-0514 [68]	92.3	80.6	92.0	87.9	88.2
Self-taught Evaluator [96]	96.6	84.2	91.5	81.0	88.3
Standard Reward Models					
RM [97]	98.3	74.5	88.0	83.8	86.4
Cohere-0514	96.4	71.3	92.3	97.7	89.4
Llama3-70B-SteerLM-RM [84]	91.3	80.3	90.6	92.8	88.8
URM-Llama3-8B [75]	96.9	78.7	95.7	88.2	89.9
ArmoRM-Llama3-8B-v0.1 [98] †	96.9	76.8	97.3	90.5	90.4
Pair-preference-model-LLaMA3-8B [63] †	98.3	65.8	94.7	89.7	87.1
Internlm2-20B-reward [99]†	98.9	76.5	95.8	89.5	90.2
DORM	95.6	83.4	93.1	89.8	90.5

3.5.2 Main Experiments on RewardBench

Table 3.1 presents the main results on RewardBench, we summarize the findings below.

- **Learn to weigh preference data helps:** DORM outperforms the best LLM-as-a-judge baseline by $+2.2\%$ in the overall RewardBench score, demonstrating that incorporating preference data weighting leads to better reward modeling accuracy than direct response ranking by large LLMs. Furthermore, DORM achieves performance on par with the strongest standard reward model baselines (e.g., *ArmoRM-Llama3-8B-v0.1*), showing that our weighting strategy effectively optimizes the use of available preference data without requiring significantly larger model capacity or data volume.

- **Significant Data Efficiency:** While achieving comparable performance to the strongest reward model baselines, our approach uses only 50k preference data, which is $10\times$ less than *ArmoRM-Llama3-8B-v0.1* and *Internlm2-20B-reward* $40\times$ less than *Internlm2-20B-reward*. This demonstrates the data efficiency of our approach. We further analyze the data efficiency in Appendix section I.2.

- **Improvements on Challenging Subtasks:** DORM exhibits notable improvements on Reasoning ($+1.5\%$) compared to the best LLM-as-a-judge baseline and Chat_Hard ($+3.1\%$) compared to the best RM baseline. This suggests that our data weighting strategy effectively prioritizes informative and reliable preference data, enhancing model performance on difficult tasks.

3.5.3 Component Effectiveness

We study the effectiveness of each component in our two-stage method. Specifically, we evaluate whether applying the prior weights and the bilevel optimization alone can improve the performance, and if the integration of them can bring extra benefits. As shown in Table 3.2, we compare the entire method with the direct SFT baseline, the method with only prior weights, and the method of bilevel optimization without prior weights. We found that

Table 3.2: Study of component effectiveness. For methods involving bilevel optimization, we have 3.5k validation data for the upper-level optimization. For a fair comparison, we incorporate the validation data into the training set for the SFT method and the method with prior weights only.

Method	**Reward-Bench Subsets**				
	Chat	**Chat Hard**	**Safety**	**Reasoning**	**Avg**
SFT	90.7	72.8	86.7	91.1	85.3
Prior weights only	91.9	77.0	85.2	91.6	86.4
Bilevel w/o prior	93.2	81.3	89.8	92.2	89.1
DORM	94.7	84.8	88.5	92.9	90.2

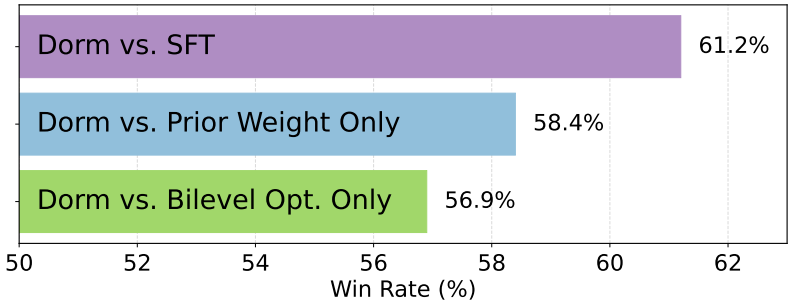


Figure 3.3: Policy model alignment results.

both prior weighting and bilevel optimization individually enhance the results over the simple SFT approach. Applying only prior weights leads to a noticeable improvement, especially on the subtask of chat hard, and employing bilevel optimization without prior weights yields a substantial gain on subtasks of chat hard and safety. Moreover, incorporating both prior weights and bilevel optimization together further boosts the performance, achieving the highest overall score. This indicates that the two components are complementary, and their combination yields better outcomes than using either alone.

3.5.4 Policy Model Alignment Results

We also evaluate the policy model aligned by DPO, where the preference data is generated by our trained RMs. In this set of experiments, we utilize the UltraFeedback [101] dataset for alignment. We divide its original 64k samples into three subsets: 20k for supervised fine-tuning, 40k for DPO, and 4k for testing. We use a Mistral-7B model as the policy

model. The alignment process consisted of one epoch of supervised fine-tuning followed by two epochs of DPO. To evaluate the win rate, we employ an independent reward model pair-preference-model-LLaMA3-8B [63], providing an unbiased assessment of our policy model's effectiveness.

Figure 3.3 shows DORM outperforms the baseline models in terms of win rate when aligning the policy model. The consistent performance advantage across both reward modeling and alignment tasks indicates that DORM not only improves reward modeling accuracy but also enhances policy alignment - the primary objective of reward modeling. This end-to-end improvement suggests that the enhanced reward signals produced by DORM translate effectively into better-aligned policies.

3.5.5 Track Data Weights Change

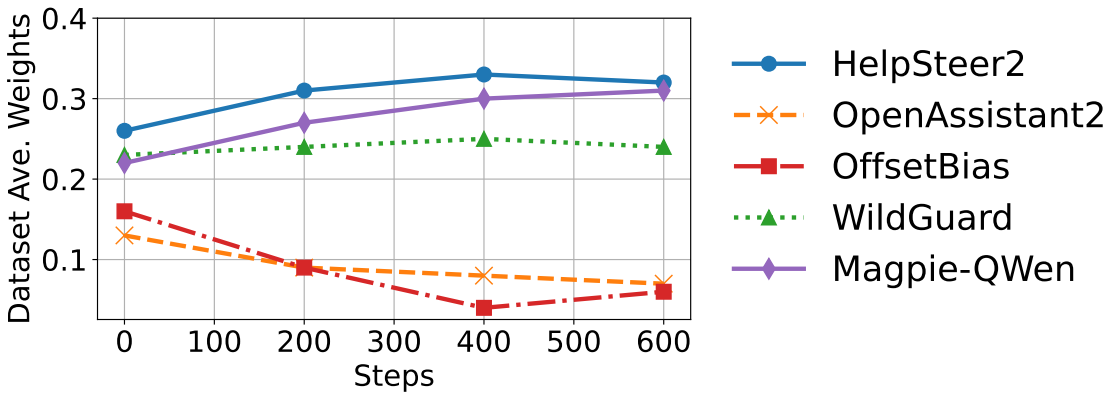


Figure 3.4: Data weights tracking in the process of reward model training.

Figure 3.4 demonstrates the data weights dynamics by tracking the average weights assigned to studied datasets. HelpSteer2 consistently receives the highest weights, increasing from 0.26 to 0.32, suggesting DORM identifies it as the most informative and reliable dataset. Magpie-QWen-2.5 shows moderate importance with gradual weight increase from 0.22 to 0.31, while WildGuard maintains relatively stable weights around 0.24. OpenAssistant2 and OffsetBias receive decreasing weights (0.13 to 0.07 and 0.16 to 0.06), indicating DORM identifies them as noisy or less reliable data and down-weights them based on the validation

loss. These varying trajectories demonstrate DORM's capability to automatically adjust data importance based on their contribution to the learning objective.

3.6 Conclusions

In this study, we presented DORM to tackle the critical issue of preference data quality estimation in aligning large language models. By integrating epistemic uncertainty with a disagreement measure, we developed a method to assess the informativeness and reliability of each preference data point. By utilizing these quality estimates as prior weights and refining them through a bilevel optimization framework, we balance prior knowledge with data-driven insights and enhance the robustness of reward models to handle diverse preference data. DORM achieves high performance with significantly less data, leading to more robust reward models and better-aligned policy models. This work highlights the importance of data quality in model alignment and provides a promising avenue for developing more reliable and human-aligned language models.

Part II

Model Efficiency in the Deployment Stage

CHAPTER 4
ENHANCING MODEL PARAMETER EFFICIENCY VIA PERTURBED KNOWLEDGE DISTILLATION

4.1 Introduction

Knowledge distillation (KD) is a widely-used technique to transfer knowledge from a large teacher model into a much smaller student model with minimum sacrifice of teacher model's predictive power [103, 104]. The vanilla training objective in KD such as KL loss [104, 105, 106] encourages the student's outputs to be close to the teacher's outputs as much as possible. This objective implicitly assumes the teacher's output distribution on the distillation data is perfectly aligned with the ground truth distribution. However, in many applications, this assumption does not hold and the teacher's output distributions can be biased from the ground truth due to various factors, such as the inductive bias encoded in the teacher model architecture [107], miscalibration in the training procedure [105], or the bias in the teacher model training set [108, 109]. Enforcing the student to blindly imitate the teacher's outputs can make the student inherit such biases and produce suboptimal predictions.

To overcome this challenge, one common approach [104] suggests scaling the teacher's logits via a temperature parameter. A proper temperature value can enhance the quality of the teacher model's output distribution by making it closer to the true label distribution [105]. However, the shifting space offered by temperature scaling is limited, and the optimal temperature value relies on resource-intensive grid search. Along a separate line, label smoothing [110] is proposed to regularize the neural networks, and modulated loss functions [111, 112] are designed to address various statistical issues in model training such as overfitting and data imbalance. Despite their potential, there is a lack of work that explores tailoring such techniques for more robust knowledge distillation.

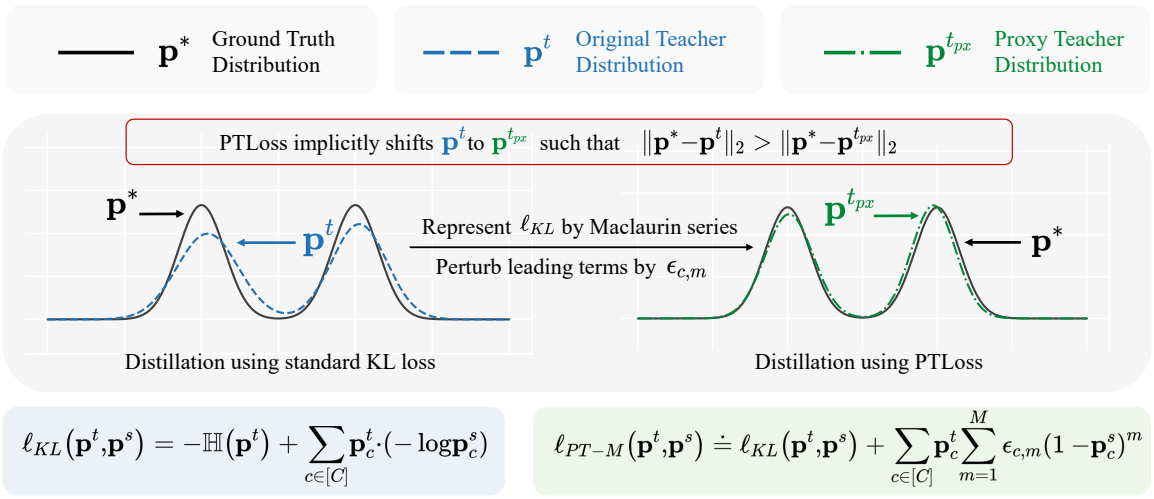


Figure 4.1: PTLoss implicitly transforms the original teacher into a proxy teacher with a distribution closer to the ground truth distribution. This approach addresses the issue of sub-optimal student models resulting from discrepancies between the teacher's output distribution and the ground truth distribution. By introducing perturbation to standard KL loss represented by its Maclaurin series, we obtain a better proxy teacher, which leads to a more effectively distilled student.

In this study, we propose *PTLoss* for knowledge distillation, which generalizes the vanilla KL loss function and implicitly creates a debiased teacher distribution closer to the ground truth. As shown in Figure 4.1, our approach does not merely mimic the original teacher model's output distribution, denoted as $\mathbf{p}^t$, but rather aims to bring the distilled model closer to the ground truth distribution $\mathbf{p}^*$, by generating a proxy teacher distribution $\mathbf{p}^{t_{px}}$. This is achieved by the proposed PTLoss, which implicitly transforms the original teacher (the dashed blue curve in Figure 4.1) into a proxy teacher (the dash-dot green curve in Figure 4.1). The resultant proxy teacher distribution is closer to the ground truth than the original teacher's output distribution, thus enhancing the distillation process.

Compared to the standard KL loss, PTLoss refrains from forcing an out-and-out imitation of the original teacher model. Instead, PTLoss moderates the distillation objective by adding perturbations to the standard KL loss. Specifically, we first decompose the standard KL loss into its Maclaurin series and then perturb its leading-order terms to construct a more flexible learning objective. This manipulation enables consequential adjustments to the teacher's output distribution. The perturbations are not arbitrary but are meticulously

calculated. To determine the perturbation extent, we compute the equivalent distribution of this implicitly shifted teacher's output distribution after perturbations (named "*proxy teacher*") and measure the empirical deviation between the proxy teacher and the ground truth data. It leads to a systematic searching strategy for the perturbation coefficients — the near-optimal perturbation coefficients should minimize the deviation between the distillation risk and the population risk on the validation set.

Theoretically, we justify the effectiveness of PTLoss by proving that it can reduce the deviation from the distillation risk compared to KL loss. We draw a connection between the PTLoss and other perturbation methods (*e.g.,* temperature scaling [104], label smoothing [110], and focal loss [111]). We illustrate that the PTLoss can debias the teacher to produce higher-fidelity outputs via a finer-grained perturbation, while subsuming existing perturbation techniques as special cases. Experiments on multiple datasets with different-sized teacher models demonstrate the empirical advantages of PTLoss.

In summary, we make the following contributions:

1. A new knowledge distillation loss function PTLoss, which formulates the vanilla KD loss in the form of Maclaurin series and perturbs it to improve the fidelity of teacher models;

2. A principled method to compute the proxy teacher for determining the perturbation coefficients in PTLoss;

3. Theoretical analysis on why PTLoss can lower the distillation risk bound and how it subsumes other perturbation methods;

4. Comprehensive experiments on multiple language understanding tasks with different-sized teacher models showing the advantage of PTLoss.

4.2 Related Work

4.2.1 Knowledge Distillation

Knowledge distillation was first proposed in [103] to compress the large models to smaller, faster models without a significant performance drop. [104] generalized this technique by introducing a temperature parameter to smooth the teacher model prediction. [113] employed contrastive learning to train the student model. Later, [114] explored the connection between KD and label smoothing, [115] and [116] studied the feeding mechanism of the teacher's knowledge. [117] decoupled the classical loss to target classes and non-target classes for KD efficiency and flexibility. [118] investigated supervisory signals and proposed to average teacher outputs for KD stability, while [119] evolves the teacher model with the student feedback in a meta learning framework. Additionally, there is also significant research on applying KD to text generation tasks, as demonstrated by [120, 121, 122, 123, 124]. However, the exploration of KD in text generation represents a distinct and extensive area of study, it is beyond the scope of this paper because our focus is on KD within language understanding tasks.

4.2.2 Distillation Theory

Concurrent with the empirical success of knowledge distillation, numerous works aim to understand its mechanisms. [104] suggest that teacher's soft labels offer "dark knowledge" through weights on incorrect labels. [105] present a statistical view, observing that a good teacher model should be Bayesian to reduce the student objective variance. [106] highlight discrepancies between teacher and student output distributions and emphasize the optimization challenge in distillation. While more recent studies [125, 126, 127, 128] explore distillation from several various angles, a gap remains between the theoretical analysis and the improved distillation techniques.

4.2.3 Loss Function Design

Our work also relates to loss function design. [111] propose a modification of the cross-entropy loss function, reshaping it to focus more on hard examples and address issues of data imbalance. [112] expand cross-entropy loss and focal loss into a linear combination of polynomial functions, primarily studying Poly-1 formulation on computer vision tasks while avoiding issues with high-order polynomial hyper-parameter searches. TaylorGLO [129] utilizes Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to optimize multivariate Taylor parameterization of a loss function and learning rate schedule. However, it lacks a principled analysis of the performance gains following perturbations. In contrast, our work provides both theoretical and empirical evidence supporting the necessity of incorporating perturbations into the Knowledge Distillation (KD) learning objective, especially when employing a high-fidelity teacher for effective student supervision.

4.3 Preliminaries

Multi-class Classification. In a multi-class classification problem with C classes, we are given a set of training examples $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ where input $x_n \in X$ and output y_n is a one-hot vector in $Y = \{y | y \in \{0,1\}^C, \mathbf{1}^T y = 1\}$ indicating the target label of example x_n. The goal is to learn a probability predictor $\mathbf{p} : X \to \mathbb{R}^C$ by optimizing the below minimal *risk*:

$$R(\mathbf{p}) = \mathbb{E}_{(x,y)}[\ell(y, \mathbf{p}(x))]. \tag{4.1}$$

where $\ell(y, \mathbf{p}(x))$ is the loss of predicting $\mathbf{p}(x)$ when the true label of example x is y.

A canonical loss function is the cross-entropy loss: $\ell_{CE}(y, \mathbf{p}(x)) = -y \log(\mathbf{p}(x))$ and we may further approximate the above risk via the *empirical risk* on the training set $\mathcal{D}$:

$$\hat{R}(\mathbf{p}; \mathcal{D}) \doteq \frac{1}{N} \sum_{n=1}^{N} y_n(-\log(\mathbf{p}(x_n))) \tag{4.2}$$

Our Problem Formulation. In this work, we study the knowledge distillation problem where the labeled training set $\mathcal{D}$ is inaccessible[1]. Specifically, we are only given an unlabeled distillation set $\mathcal{D}_u$, a teacher model $\mathbf{p}^t$, and asked to learn a student model $\mathbf{p}^s$.

Standard Distillation Strategy. A standard KD strategy [104] is to replace the ground truth one-hot label y_n in Eq. Equation 4.2 with the teacher model's output probabilistic label estimate $\mathbf{p}^t(x_n)$ and utilize the KL divergence loss to learn the student model $\mathbf{p}^s$ via the *distillation empirical risk*:

$$\tilde{R}_{KL}(\mathbf{p}^s; \mathbf{p}^t, \mathcal{D}_u) \doteq \frac{1}{N_u} \sum_{n=1}^{N_u} \ell_{KL}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right), \tag{4.3}$$

where $N_u = |\mathcal{D}_u|$ and $\ell_{KL}(\mathbf{p}, \mathbf{q}) = KL(\mathbf{p}||\mathbf{q}) = \mathbf{p}^T \log(\mathbf{p}) - \mathbf{p}^T \log(\mathbf{q})$.

4.4 Perturbed Distillation Loss

Using the KL divergence loss (in short "KL loss") for distillation essentially assumes the teacher model is perfect and forces the student model to mimic the teacher's output label distribution. In reality, the teacher model can produce a biased estimate of label distribution and lead to a sub-optimal student model, as demonstrated by both theoretical analysis [105] and empirical observations [131] (as well as our experiments in Section subsection 4.6.1).

In this work, we present a new distillation loss that generalizes the standard KL loss to accommodate various degrees of distribution gaps between the biased teacher's output distribution and the underlying ground truth distribution. Inspired by the PolyLoss [112], we propose to first replace the logarithmic terms in the standard KL loss with their corresponding

[1]This setting reflects the real-world scenario where large teacher models (e.g., ChatGPT [130] and GPT4 [1]) only expose their outputs and/or APIs without original training data because of their large model sizes and cautions toward data leakage/misuse.

Maclaurin series and then perturb the polynomial terms as follows:

$$\log(x) = -\sum_{m=1}^{\infty} \frac{(1-x)^m}{m} \quad \xrightarrow[\text{term coefficients}]{\text{Perturb polynomial}} \quad \log(x) \approx -\sum_{m=1}^{\infty} \left(\frac{1}{m} + \epsilon_m\right)(1-x)^m. \tag{4.4}$$

Here, we essentially replace the original coefficient $\frac{1}{m}$ of the m-th order polynomial term in the standard KL loss to $(\frac{1}{m} + \epsilon_m)$. By further replacing the logarithmic terms in standard KL loss (Eq. Equation 4.3) with the above Eq. Equation 4.4, we will have:

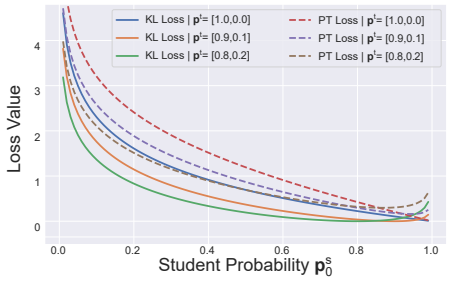
$$\begin{aligned} \ell_{KL}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right) &= -\mathbb{H}\left(\mathbf{p}^t(x_n)\right) + \sum_{c\in[C]} \mathbf{p}_c^t(x_n)[-\log \mathbf{p}_c^s(x_n)] \\ \approx -\mathbb{H}\left(\mathbf{p}^t(x_n)\right) &+ \sum_{c\in[C]} \mathbf{p}_c^t(x_n)\left[-\log \mathbf{p}_c^s(x_n) + \sum_{m=1}^{\infty} \epsilon_{c,m}(1-\mathbf{p}_c^s(x_n))^m\right], \end{aligned} \tag{4.5}$$

where $\mathbf{p}_c^t(x_n)$ and $\mathbf{p}_c^s(x_n)$ denote the probability that example x_n belongs to the class c according to the teacher (student) model, and $\mathbb{H}\left(\mathbf{p}^t(x_n)\right)$ is the entropy of the teacher output distribution.

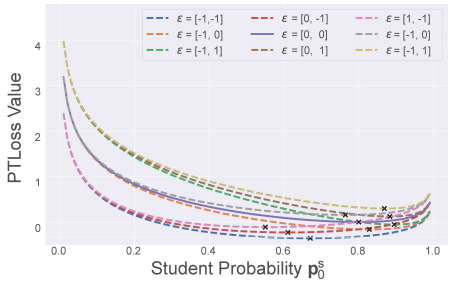
We can further separate out the perturbation coefficients on the right hand side of Eq. Equation 4.5 and merge $\sum_{c\in[C]} \mathbf{p}_c^t(x_n)\left[-\log \mathbf{p}_c^s(x_n)\right]$ with $\mathbb{H}\left(\mathbf{p}^t(x_n)\right)$ to obtain our perturbed distillation loss:

$$\begin{aligned} \ell_{PT}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right) \doteq\ & \ell_{KL}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right) \\ &+ \sum_{c\in[C]} \mathbf{p}_c^t(x_n) \sum_{m=1}^{\infty} \epsilon_{c,m}\left(1-\mathbf{p}_c^s(x_n)\right)^m. \end{aligned} \tag{4.6}$$

The above equation presents our perturbed distillation loss in its most general form. In practice, however, we cannot tune infinite number of coefficients $\epsilon_{c,m}$ and thus we propose to only tune the first M leading polynomial coefficients while keeping the rest unchanged as



(a) Loss values with different teacher probabilities $\mathbf{p}_0^t$ in $\{1.0, 0.9, 0.8\}$. For PTLoss, we fix the perturbation order as 1 and the perturbation coefficients $\epsilon = [1, 1]$. Consider the case that ground truth probability is $[1, 0]$, PTLoss adjusts the student's predictions by nudging them towards the ground truth, effectively mitigating the bias present in the teacher's output probabilities.



(b) PTLoss values with different perturbations. We fix the teacher probability $\mathbf{p}_0^t = [0.8, 0.2]$ and vary the perturbation coefficients ϵ, while the perturbation order is always fixed to 1. The black cross denotes the best student model output $\mathbf{p}_0^s$ that achieves the lowest loss value. This shows PTLoss can enable flexible adjustments to the loss curve and effectively reduces the bias of the teacher's output.

Figure 4.2: Intuitive understanding of our PTLoss for the binary classification task.

follows:

$$\ell_{PT\text{-}M}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right) \doteq \ell_{KL}\left(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)\right) + \sum_{c\in[C]} \mathbf{p}_c^t(x_n) \sum_{m=1}^{M} \epsilon_{c,m}\left(1 - \mathbf{p}_c^s(x_n)\right)^m. \tag{4.7}$$

We can see that if we set all $\epsilon_{c,m}$ to 0, the ℓ_{PT} falls back to the ℓ_{KL} and thus the perturbed distillation loss can be considered as a generalization of the standard KL loss.

Figure 4.2 presents how PTLoss adjusts biased teachers. For visualization simplicity, we set the number of classes $C = 2$. In Figure 4.2a, we vary the teacher probability to show how the biased teacher model will impact the distilled student model under either the standard KL loss or our proposed PTLoss. We observe that PTLoss can guide the student's predictions toward the ground truth and thus effectively reduces the inherent bias in the teacher's output probabilities. In Figure 4.2b, we demonstrate PTLoss enables a diverse shift space to the loss curve. By setting the perturbation coefficients, PTLoss allows flexible adjustments to the loss curve. Combining with our perturbation coefficients selection methods discussed in subsection 4.5.3, we can determine the perturbation to optimize the distillation process.

4.4.1 Connections to other perturbation methods

We establish the connections between PTLoss and other related methods that transform the teacher output probabilities, such as label smoothing [110], temperature scaling [104], and focal loss [111]. We show that the loss shift space produced by PTLoss encompasses these techniques and thus PTLoss can offer additional adjustment capabilities. We present the connections in this section and detail the derivation in Appendix E.

KL Loss. The connection between PTLoss and the standard KL loss is quite direct. As we represent the standard KL loss in Maclaurin series and add perturbations, we can easily revert PTLoss to the standard KL loss by setting all perturbation coefficients $\epsilon_{c,m}$ to 0.

Focal Loss. Focal loss incorporates a factor $(1-p)^\gamma$ in the loss function. We demonstrate PTLoss can subsume focal loss by expressing the perturbation coefficients as a function of the factor $(1-p)^\gamma$. For simplicity, we denote $\mathbf{p}^t(x_n)$ and $\mathbf{p}^s(x_n)$ as $\mathbf{p}^t$ and $\mathbf{p}^s$. By applying Focal loss to KD, we have:

$$\ell_{focal}\left(\mathbf{p}^t, \mathbf{p}^s\right) = -\mathbb{H}\left(\mathbf{p}^t\right) + \sum_{c\in[C]} \mathbf{p}_c^t (1-\mathbf{p}_c^s)^\gamma [-\log \mathbf{p}_c^s], \tag{4.8}$$

where $(1-\mathbf{p}_c^s)^\gamma$ is a factor and the parameter $\gamma > 0$ reduces the relative loss for well-classified examples.

To bridge the connection between PTLoss and the focal loss, we compare Equation 4.8 to Equation 4.5 and establish the following relationship:

$$\sum_{m=1}^{\infty} (\frac{1}{m} + \epsilon_{c,m})(1-\mathbf{p}_c^s)^m = \sum_{m=1}^{\infty} \frac{(1-\mathbf{p}_c^s)^\gamma}{m} \cdot (1-\mathbf{p}_c^s)^m, \tag{4.9}$$

which leads to the perturbation coefficients as follows:

$$\epsilon_{c,m} = \frac{(1-\mathbf{p}_c^s)^\gamma - 1}{m}. \tag{4.10}$$

By incorporating the derived perturbation coefficients $\epsilon_{c,m}$ in our proposed method, we demonstrate that PTLoss can effectively subsume the focal loss. In other words, PTLoss generalizes the focal loss and can adapt to more modulating factors $(1-p)^{\gamma}$ for handling the class imbalance problem and improving the knowledge distillation performance.

Temperature Scaling. We compare PTLoss with temperature scaling and claim that PTLoss subsumes it with appropriate approximation. As described in [104], the logits are adjusted by a temperature to control sharpness or smoothness of the probability distribution:

$$\mathbf{p}_{\tau,0} = \frac{exp(z_0/\tau)}{exp(z_0/\tau) + exp(z_1/\tau)}, \tag{4.11}$$

where τ is the temperature and z_c is the logits. Here we use binary classification (*i.e.,* $c = 0, 1$) for the derivation simplicity without loss of generality.

We denote $\mathbf{p}^t_{\tau,c}$, $\mathbf{p}^s_{\tau,c}$ as the teacher, student probability scaled by temperature τ. Incorporating temperature scaling, the KL loss can be formulated as:

$$\ell^{temp}_{KL}\left(\mathbf{p}^t_{\tau}, \mathbf{p}^s_{\tau}\right) = -\mathbb{H}\left(\mathbf{p}^t_{\tau}\right) + \sum_{c\in[C]} \mathbf{p}^t_{\tau,c} \cdot (-\log \mathbf{p}^s_{\tau,c}). \tag{4.12}$$

Similarly, to bridge PTLoss with temperature scaling, we establish equality between Equation 4.12 and Equation 4.5, this leads to

$$\epsilon_{c,m} = \frac{a}{m} \cdot (1 + (\frac{1-b}{1-\mathbf{p}^s_c})^m) - \frac{1}{m}, \tag{4.13}$$

where $a = exp(z^t_1 - z^t_0)/exp(\frac{z^t_1 - z^t_0}{\tau})$, $b = exp(z^s_1 - z^s_0)/exp(\frac{z^s_1 - z^s_0}{\tau})$. The derivation clearly shows how PTLoss, through the appropriate selection of $\epsilon_{c,m}$, effectively subsumes temperature scaling. This capability to encompass temperature scaling further underlines the versatility of our approach.

Label Smoothing. We compare PTLoss with the label smoothing method and claim

that label smoothing proposed in [110] is a special case of PTLoss. According to the implementation in [110], we can smooth the teacher labels in KD by

$$\mathbf{p}_c^{t_{ls}} = (1-\delta)\mathbf{p}_c^t + \delta/2, \tag{4.14}$$

with a smoothing parameter δ. Starting from Equation 4.5, we can replace the term $\mathbf{p}_c^t$ by its smooth version $\mathbf{p}_c^{t_{ls}}$. Then the original Equation 4.5 with label smoothing becomes:

$$\ell_{KL}^{ls}\left(\mathbf{p}^t, \mathbf{p}^s\right) = -\mathbb{H}\left(\mathbf{p}^{t_{ls}}\right) + \sum_{c\in[C]} \mathbf{p}_c^{t_{ls}} \cdot (-\log \mathbf{p}_c^s) \tag{4.15}$$

For the entropy of the teacher output, the smooth version $\mathbb{H}\left(\mathbf{p}^{t_{ls}}\right)$ is different from the original $\mathbb{H}\left(\mathbf{p}^t\right)$ with only a constant C, which can be ignored when optimizing the loss function. Similarly, by letting $\ell_{KL}^{ls}\left(\mathbf{p}^t, \mathbf{p}^s\right) = \ell_{PT}\left(\mathbf{p}^t, \mathbf{p}^s\right)$, we obtain

$$\epsilon_{c,m} = \frac{\Delta \mathbf{p}_c^t}{m\mathbf{p}_c^t} \xrightarrow{\Delta \mathbf{p}_c^t = \delta/2 - \delta \mathbf{p}_c^t} \epsilon_{c,m} = \frac{\delta}{m}\left(\frac{1}{2\mathbf{p}_c^t} - 1\right). \tag{4.16}$$

In shows the connection between the two losses can be expressed through a specific $\epsilon_{c,m}$, which depends on the smoothing parameter δ. This derivation highlights that PTLoss generalizes the label smoothing method and provides a more flexible framework that encompasses the effects of label smoothing.

In summary, we establish that PTLoss can effectively serve as a generalized framework that includes various techniques by manipulating the perturbation coefficients $\epsilon_{c,m}$. It is important to note that our goal is not to directly solve for the perturbation coefficients to make PTLoss equivalent to the alternative perturbation methods. Instead, we aim to show that our approach covers the loss shift space produced by them. To determine the real perturbation coefficient, we establish a principled searching method via the best proxy teacher, as demonstrated in the following section.

4.5 Proxy Teacher and The Principle of Selecting Polynomial Coefficients

In this section, we first present a theorem to show how the teacher model affects the gap of a student model's distillation empirical risk and its population risk (§ subsection 4.5.1). Then, we demonstrate that using PTLoss implicitly transforms the original teacher model to a *proxy teacher* under the KL loss. Based on the above theorem, we know when this proxy teacher distribution is closer to the true distribution, we will have a better distilled student model (§ subsection 4.5.2). Finally, we establish our principle of selecting the perturbation coefficients in PTLoss: searching the coefficients that lead to a proxy teacher closest to the empirical estimate of true distribution on a validation set (§ subsection 4.5.3).

4.5.1 The Connection of the Teacher Model and the Risks of Student Model

Theorem 1 *Given a teacher model $\mathbf{p}^t$, an unlabeled distillation dataset $\mathcal{D}_u$ with an unknown true distribution $\mathbf{p}^*$, we have for any probability predictor $\mathbf{p} : \mathcal{X} \to \mathbb{R}^C$:*

$$\mathbb{E}\left[(\tilde{R}_{KL}(\mathbf{p}; \mathbf{p}^t, \mathcal{D}_u) - R(\mathbf{p}))^2\right] \leq \frac{2}{N_u} \cdot \mathbb{V}\left[\mathbf{p}^t(x)^T \log(\mathbf{p}(x))\right] + \mathcal{O}\left(\left(\mathbb{E}_x\left[\|\mathbf{p}^t(x) - \mathbf{p}^*(x)\|_2\right]\right)^2 + \mathbb{E}_x\left[\left(\mathbf{p}^t(x)^T \log \mathbf{p}^t(x)\right)^2\right]\right),$$

where $\mathbb{V}[\cdot]$ denotes the variance of a random variable.

We defer the detailed proofs of above theorem to Appendix D and focus on its implications here. We can see that the gap between a model $\mathbf{p}$'s distillation empirical risk and its population risk depends on three terms: (1) the variance of its KL distance to the teacher model $\mathbf{p}^t$, (2) the L_2 distance between the teacher model output distribution $\mathbf{p}^t$ and the true distribution $\mathbf{p}^*$, and (3) the entropy of the teacher distribution. In practice, obtaining a sizable unlabeled distillation set $\mathcal{D}_u$ is relatively straightforward, which leads to a large value of N_u. As a result, the first term (of order $O(1/N_u)$) will converge to 0 as N_u keeps

increasing and the latter two terms (one quantifies the distance between teacher $\mathbf{p}^t$ and true $\mathbf{p}^*$, and the other quantifies the teacher's uncertainty) will dominate the risk gap. This observation also resonates with our intuition that an accurate and well-calibrated teacher yields better improved bounds on the generalization error of the student.

4.5.2 The Equivalence of Proxy Teacher under KL Loss and Original Teacher under PTLoss

The above theorem states that an ideal teacher model, when used in KL loss for distillation, should output a distribution as close to the true distribution as possible. In reality, however, the teacher model is usually fixed. Here, we show that using PTLoss for distillation can implicitly transform the original teacher to a *proxy teacher* under the KL loss. Namely, given the original teacher model $\mathbf{p}^t$ and a set of perturbation coefficients $\{\epsilon_{c,m}\}$ in PTLoss, we can obtain a proxy teacher $\mathbf{p}^{t_{px}}$ such that:

$$\begin{aligned}\tilde{R}_{KL}(\mathbf{p}^s; \mathbf{p}^{t_{px}}, \mathcal{D}_u) &= \tilde{R}_{PT\text{-}M}(\mathbf{p}^s; \mathbf{p}^t, \mathcal{D}_u) \\ &= \frac{1}{N_u}\sum_{n=1}^{N_u} \ell_{PT\text{-}M}(\mathbf{p}^t(x_n), \mathbf{p}^s(x_n)),\end{aligned} \tag{4.17}$$

which establishes the equivalence of proxy teacher under KL loss and original teacher under PTLoss. With the proxy teacher $\mathbf{p}^{t_{px}}$, we aim to determine the best perturbation coefficients $\{\epsilon_{c,m}\}$. Note for each $\{\epsilon_{c,m}\}$, we can obtain a proxy teacher. We illustrate how we obtain the proxy teacher in the rest of this subsection, and discuss how to select the best perturbation coefficients in § subsection 4.5.3.

Intuitively, the proxy teacher is derived by solving the below optimization problem:

$$\min_{\mathbf{p}^{t_{px}}} \quad \|\tilde{R}_{PT\text{-}M}(\mathbf{p}^s; \mathbf{p}^t, \mathcal{D}_u) - \tilde{R}_{KL}(\mathbf{p}^s; \mathbf{p}^{t_{px}}, \mathcal{D}_u)\|_2. \tag{4.18}$$

In practice, however, we do not need the above risk equivalence in Eq. Equation 4.17 to hold for all possible student models $\mathbf{p}^s$. Instead, we focus on the minimizer of the left-hand side of Eq. Equation 4.17 because it is practically close to the final learned student model. By substituting this minimizer $\mathbf{p}^s = \mathbf{p}^{t_{px}}$ into Eq. Equation 4.18, the second term in the norm of Eq. Equation 4.18 becomes 0, and the first term could be expanded by its definition in Eq. Equation 4.17, we thus have the following objective:

$$\min_{\mathbf{p}^{t_{px}}} \Bigg\| \frac{1}{N_u} \sum_{n=1}^{N_u} \Bigg(\ell_{KL}(\mathbf{p}^t(x_n), \underline{\mathbf{p}^{t_{px}}(x_n)}) + \sum_{c\in[C]} \mathbf{p}_c^t(x_n) \sum_{m=1}^{M} \epsilon_{c,m} (1 - \underline{\mathbf{p}_c^{t_{px}}(x_n)})^m \Bigg) \Bigg\|_2 . \tag{4.19}$$

This objective enables us to solve $\mathbf{p}^{t_{px}}$ given $\mathbf{p}^t$ and $\{\epsilon_{c,m}\}$, where $\mathbf{p}^t$ is the teacher's output probability on the validation set, and $\{\epsilon_{c,m}\}$ is a given set of perturbation coefficients. However, this optimization problem is nonlinear and lacks a closed-form analytical solution. Consequently, we compute the $\mathbf{p}^{t_{px}}$ using the numerical approach[2]. Specifically, the optimization problem defined in Eq. Equation 4.19 is solved via the algorithm 'scipy.optimize.fsolve', which is a hybrid method of the Newton-Raphson method and the Levenberg-Marquardt algorithm. For better numerical stability, we actually solve the equation in logit space (instead of the vanilla probability space) and use softmax function to map it back to the final probability. Another advantage of this approach is that we remove the probability constraint of $\mathbf{p}^{t_{px}}$. We also input the analytical form of the Jacobian of our optimization objective into the solver (via the 'fprime' parameter) and set the initial estimate of $\mathbf{p}^{t_{px}}$ to be the original teacher $\mathbf{p}^t$ (via the 'x0' parameter). For all the other parameters in 'scipy.optimize.fsolve', we use their default values.

We have also considered an alternative solution to this optimization problem, which

[2]We use a hybrid algorithm of the Newton-Raphson method and the Levenberg-Marquardt algorithm as defined in 'scipy.optimize.fsolve' https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html.

Algorithm 2 Automated Perturbation Coefficients Selection

Require: Validation set $\mathcal{D}_v$, Teacher model $\mathbf{p}^t$, Max perturbation order N_M, Max search trails N_k, Perturbation coefficient search space $\mathcal{S}$.
Initialize $\{\epsilon^*_{c,m}\} \leftarrow \{\}$, $\hat{Q}^* \leftarrow \infty$.
for $M = 1$ to N_M **do**
  for $k = 1$ to N_k **do**
    Randomly sample a perturbation coefficients set $\{\epsilon_{c,m}\} \in \mathcal{S}$.
    Solve the proxy teacher $\mathbf{p}^{t_{px}}$ given $\{\epsilon_{c,m}\}$ and $\mathbf{p}^t$ via Eq. Equation 4.19.
    Compute the quality score of perturbation coefficients $\hat{Q}(\{\epsilon_{c,m}\})$ via Eq. Equation 4.20.
    if $\hat{Q}(\{\epsilon_{c,m}\}) < \hat{Q}^*$ **then**
      $\hat{Q}^* \leftarrow \hat{Q}(\{\epsilon_{c,m}\})$, $\{\epsilon^*_{c,m}\} = \{\epsilon_{c,m}\}$.
    end if
  end for
end for
return $\{\epsilon^*_{c,m}\}$.

involves defining a parameterized function $g_\theta(\cdot) : [0,1]^C \rightarrow [0,1]^C$ that explicitly transforms the original teacher to the proxy teacher, namely $g_\theta(\mathbf{p}^t) = \mathbf{p}^{t_{px}}$. We would then find the best θ minimizing the above objective (possibly via gradient-based methods). This approach leads to a smooth proxy teacher but also introduces bias from the function class defined by θ. Therefore, we leave it to future work and resort to the numerical approach in this study.

4.5.3 Selecting Perturbation Coefficients via the Best Proxy Teacher

For each candidate set of perturbation coefficients $\{\epsilon_{c,m}\}$ in PTLoss, we can find a corresponding proxy teacher and compute its risk deviation upper bound according to theorem Theorem 1. In practice, the size of distillation set N_u is typically large and thus we can omit the $O(1/N_u)$ variance term. Furthermore, since the ground truth distribution $\mathbf{p}^*$ is unknown, we use an unbiased estimator to replace it. Finally, we replace the expectation by the sample mean and define the empirical risk below:

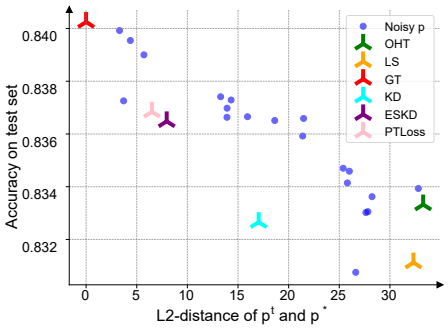
$$\begin{aligned}\hat{Q}(\{\epsilon_{c,m}\}) = &\left(\frac{1}{N_v}\sum_{n=1}^{N_v}\left[\|\mathbf{p}^{t_{px}}(x_n) - \mathbf{y}_n\|_2\right]\right)^2 \\ &+ \frac{1}{N_v}\sum_{n=1}^{N_v}\left[\left(\mathbf{p}^{t_{px}}(x_n)^T \log \mathbf{p}^{t_{px}}(x_n)\right)^2\right],\end{aligned} \tag{4.20}$$

where N_v is the size of validation set and $\mathbf{y}_n$ is a one-hot label vector of x_n, serving as the unbiased estimation of $\mathbf{p}^*(x_n)$. We use $\hat{Q}(\{\epsilon_{c,m}\})$ as a "quality score" for each candidate coefficients set. Users can define a search space of $\{\epsilon_{c,m}\}$ and we will pick the optimal $\{\epsilon^*_{c,m}\}$ that minimizes $\hat{Q}$. We present the pseudo-code for selecting perturbation coefficients in Algorithm 2, and the search time for perturbation coefficients is detailed in subsection C.0.3.

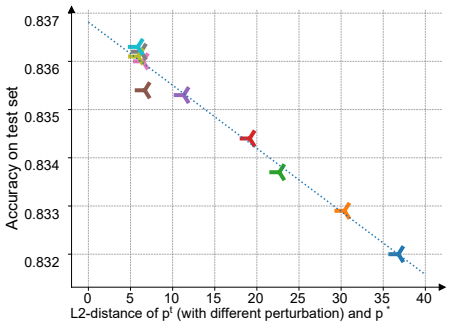
4.6 Experiments

In this section, we first conduct experiments on a synthetic dataset to verify our assumption that the teacher outputting a distribution closer to the ground truth distribution leads to a better student (§subsection 4.6.1). Then, we present our main results on 6 real-world NLP datasets (§subsection 4.6.2). Moreover, we show how the proxy teacher enhances the distillation process(§subsection 4.6.3). In the appendix, we further evaluate the performance of PTLoss on CIFAR-100 to show its potential in computer vision tasks (§subsection 4.6.4).

4.6.1 Experiments on Synthetic Gaussian Dataset



(a) OHT - one-hot training; LS - label smoothing; GT - ground truth; KD - knowledge distillation; ESKD: early-stopped KD [118]; PTLoss: here we use 3-order perturbation.



(b) Correlation between the L_2-distance (between the teacher model $\mathbf{p}^t$ with different levels of perturbations and the ground truth $\mathbf{p}^*$) and the test accuracy of the student model.

Figure 4.3: Experiments on a synthetic Gaussian dataset.

We first conduct an illustrative experiment with a synthetic dataset where the ground truth distribution $\mathbf{p}^*(x)$ is known. Specifically, we follow [118] to generate 10^5 examples

from a mixture of Gaussian distribution and train an MLP with 3 hidden layers on this synthetic dataset. This is a 3-class toy Gaussian dataset with 10k data points, divided into training, validation, and test sets with a split ratio $[0.9, 0.05, 0.05]$. The underlying model in this set of experiments is a 2-layer MLP with ReLU activation, and the hidden size is 128 for each layer. We set the learning rate as 5×10^{-4}, the batch size as 32, and the training epochs as 100.

The sampling process is implemented as follows: we first choose the label y using a uniform distribution across all the 3 classes. Next, we sample $x|_{y=k} \sim \mathcal{N}(\mu_k, \sigma^2 I)$ as the input signal. Here $\sigma = 2$ and μ_k is a 30-dim vector with entries randomly selected from $\{-1, 0, 1\}$.

We compare PTLoss with 4 baselines: one-hot supervision (OHT), label smoothing (LS), standard knowledge distillation (KD), and early-stopped knowledge distillation (ESKD) (see details in below § subsection 4.6.2). As illustrated in Figure 4.3a, the quality of the distilled student improves as the L_2-distance between the teacher distribution and the ground truth distribution decreases. On this synthetic Gaussian dataset, PTLoss also outperforms the baselines after adding a 3-order perturbation.

In Figure 4.3b, we sample 10 proxy teachers in different stages of the perturbation coefficient searching process (§subsection 4.5.3) and compare their results. It is clear that a teacher model with a smaller L_2-distance to the ground truth distribution can lead to a better student model. This observation verifies our hypothesis in Eq. Equation 4.20 — searching a proxy teacher closer to the ground truth distribution can reduce the empirical deviation and improve the distilled student model.

4.6.2 Experiments on Natural Language Datasets

Tasks and Datasets. We conduct our main experiments on 6 natural language datasets, including:

1. CoLA [132] for linguistic acceptability;

2. MNLI [133] for multi-genre natural language inference;

3. MRPC [134] for paraphrase similarity matching;

4. RTE [135] for textual entailment inference;

5. SST-2 [135] for sentiment analysis;

6. BoolQ [136] for boolean question answering.

We list the detailed dataset statistics in the subsection C.0.1.

Model Architectures. For the teacher model, we choose the T5 architecture [137] and select two teacher models of different scales. Specifically, we use T5-xxl with 11 billion parameters and T5-large with 770 million parameters. For the student model, we use the BERT-base model [138] with 110 million parameters.

Table 4.1: Main results on natural language datasets. The student model (BERT-base) is distilled from teacher models of different sizes (T5-xxl and T5-large). All results are averaged over three runs. The bolded numbers indicate the best results, while the underscore "_" denotes the second-best results.

Method	CoLA (Matt.)	MNLI (Acc.)	MRPC (F1)	RTE (Acc.)	SST-2 (Acc.)	BoolQ (Acc.)	Average
Teacher T5-xxl	71.5	94.7	92.4	92.2	96.4	89.1	89.4
Standard KL	58.8 ± 0.3	90.3 ± 0.2	88.3 ± 0.4	78.1 ± 0.1	89.2 ± 0.3	69.5 ± 0.2	79.0
Temp. Scaling	59.4 ± 0.3	90.7 ± 0.1	88.9 ± 0.3	79.6 ± 0.3	89.4 ± 0.2	$\underline{72.0} \pm 0.3$	80.0
Label Smoothing	59.3 ± 0.6	90.6 ± 0.4	89.2 ± 0.7	79.2 ± 0.4	89.9 ± 0.3	68.6 ± 0.4	79.6
Focal	59.2 ± 0.4	90.7 ± 0.3	88.7 ± 0.9	80.4 ± 0.4	89.3 ± 0.3	68.2 ± 1.5	79.6
Flooding	58.9 ± 0.5	90.6 ± 0.4	89.6 ± 0.6	80.2 ± 0.7	89.3 ± 0.4	69.3 ± 0.5	79.7
CRD	59.5 ± 0.5	90.5 ± 0.3	90.6 ± 0.4	81.1 ± 0.2	$\underline{89.6} \pm 0.3$	71.8 ± 0.6	80.5
Annealing KD	59.8 ± 0.3	90.7 ± 0.3	90.0 ± 0.5	80.7 ± 0.2	89.3 ± 0.5	70.7 ± 0.5	80.2
FilterKD	59.2 ± 0.4	90.7 ± 0.2	89.5 ± 0.3	80.4 ± 0.3	89.3 ± 0.2	69.6 ± 0.9	79.8
MetaDistill	$\underline{60.4} \pm 0.2$	$\underline{90.8} \pm 0.3$	$\mathbf{91.4} \pm 0.4$	$\underline{81.3} \pm 0.1$	89.5 ± 0.2	71.9 ± 0.7	$\underline{80.9}$
(ours)	$\mathbf{61.2} \pm 0.3$	$\mathbf{91.1} \pm 0.1$	$\underline{91.2} \pm 0.3$	$\mathbf{83.5} \pm 0.2$	$\mathbf{90.3} \pm 0.1$	$\mathbf{73.1} \pm 0.5$	**81.8**
Teacher T5-large	61.4	93.6	92.1	87.2	95.5	77.9	84.6
Standard KL	54.8 ± 0.2	90.0 ± 0.1	87.8 ± 0.3	77.6 ± 0.2	88.8 ± 0.1	69.5 ± 0.2	78.1
Temp. Scaling	55.6 ± 0.3	90.4 ± 0.1	88.7 ± 0.2	79.4 ± 0.2	89.2 ± 0.5	70.4 ± 0.6	79.1
Label Smoothing	56.4 ± 0.4	90.6 ± 0.2	89.2 ± 0.6	79.2 ± 0.4	89.2 ± 0.4	69.1 ± 1.2	79.1
Focal	56.0 ± 0.2	90.3 ± 0.1	88.4 ± 0.5	79.9 ± 0.4	89.3 ± 0.5	68.9 ± 0.5	78.8
Flooding	57.8 ± 0.3	90.0 ± 0.6	89.5 ± 0.4	79.5 ± 0.4	89.0 ± 0.4	68.9 ± 0.6	79.3
CRD	58.2 ± 0.3	90.2 ± 0.4	$\underline{89.8} \pm 0.3$	80.3 ± 0.2	$\underline{89.4} \pm 0.4$	$\underline{70.5} \pm 0.5$	79.7
Annealing KD	58.3 ± 0.2	90.4 ± 0.3	$\underline{89.8} \pm 0.5$	79.9 ± 0.1	$\underline{89.4} \pm 0.4$	69.7 ± 0.4	79.6
FilterKD	56.7 ± 0.3	90.2 ± 0.2	89.1 ± 0.4	78.8 ± 0.2	89.2 ± 0.3	69.2 ± 0.6	78.9
MetaDistill	$\underline{58.6} \pm 0.2$	$\mathbf{90.7} \pm 0.3$	89.6 ± 0.1	$\underline{81.0} \pm 0.1$	89.3 ± 0.2	70.4 ± 0.2	$\underline{80.1}$
(ours)	$\mathbf{60.5} \pm 0.2$	$\mathbf{90.7} \pm 0.1$	$\mathbf{91.1} \pm 0.4$	$\mathbf{82.7} \pm 0.1$	$\mathbf{90.0} \pm 0.2$	$\mathbf{71.0} \pm 0.3$	**81.0**

Compared Methods. We compare our proposed PTLoss with the following baselines:

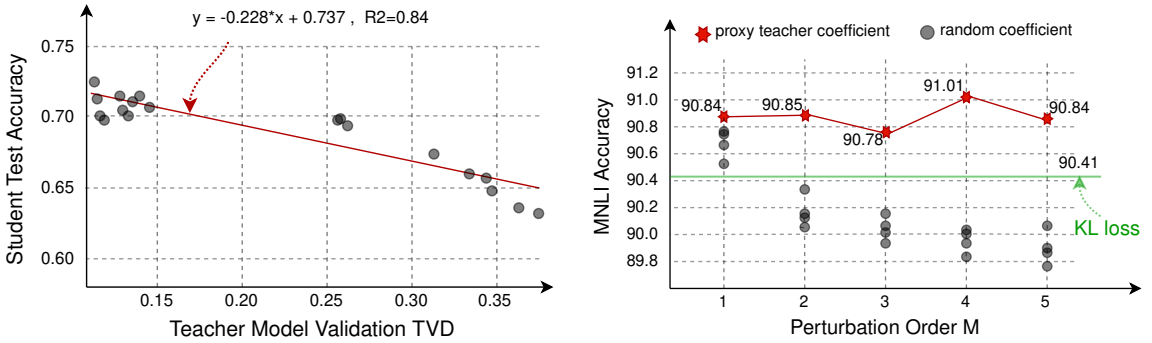
1. Standard KL loss [139]: adopts standard KL divergence loss for knowledge distillation;
2. Temperature scaling [104]: scales the teacher output logits via a temperature hyper-parameter;
3. Label smoothing [110]: smooths the teacher output class probabilities by a small scalar;
4. Focal loss [111]: modulates the cross-entropy loss to focus on hard examples;
5. Flooding [140]: a regularization method to intentionally prevent further reduction of the training loss;
6. CRD [113]: uses a contrastive objective in knowledge distillation;
7. AnnealingKD [115]: feeds the rich information provided by the teacher's soft-targets incrementally;
8. FilterKD [118]: trains the student from the smoothed predictions of the teacher network;
9. MetaDistill [119]: evolves the teacher network with the feedback from the distilled student in a meta learning framework.

For all the baselines, we conduct an exhaustive hyper-parameter search on the validation set. For our own PTLoss method, we set its perturbation order $M = 5$ and use the proxy teacher-based method to search its perturbation coefficients (§subsection 4.5.3). See subsection C.0.2 for more details. We run each method with three different random seeds and report its average performance.

Main Results. Table 4.1 shows the main results on 6 NLP datasets. We have the following observations: (1) PTLoss outperforms on 11 out of the total 12 tasks, achieving the best average performance across the board. The only exception is MetaDistill, which tops the results on the MRPC when using the T5-xxl teacher and ties with PTLoss on MNLI when using the T5-large teacher. (2) The advantages offered by PTLoss are robust, regardless of the scale of the teacher model. Notably, as the disparity in scale between the teacher model and student model reduces, the performance gap between them also narrows. (3) In comparison to vanilla KD, which utilizes the standard KL, PTLoss showcases significant

enhancement. Specifically, it exceeds standard KL by an average of 2.8% and 2.9%, respectively. (4) Surveying the baseline methods, MetaDistill stands out, securing the second-highest performance across most tasks. On the whole, the cluster of KD methods generally outstrips the simple regularization methods.

4.6.3 Proxy Teacher Analysis



(a) Correlation between the teacher-to-ground-truth distance and the student model performance. Experiments are conducted on BoolQ.

(b) The method v.s. random search for the perturbation coefficient selection. Experiments are conducted on MNLI.

Figure 4.4: PTLoss analysis.

Correlation between teacher's distance to ground truth and student's performance. To explore where PTLoss's performance gains come from, we train multiple teacher models on the BoolQ dataset and distill them into the student models. Figure 4.4a shows the student model performance on the test set is highly correlated with the distance between the teacher model's output distribution and the ground truth distribution on the validation set. This result resonates with our findings from synthetic datasets(§ subsection 4.6.1), confirming that, on real-world datasets, a teacher model with a predictive distribution closer to the ground truth can produce a more effectively distilled student.

Effectiveness of Perturbation Coefficients Search. We continue to validate the effectiveness of the proxy teacher-based perturbation coefficients selection method using MNLI as a representative dataset. Specifically, we vary the perturbation order M from 1 to 5 and report

the performance of the student models distilled via PTLoss with different perturbation coefficients. These coefficients are obtained either by minimizing the empirical risk deviation of proxy teacher (c.f. Eq. Equation 4.19) or via random sampling from the space of $[-1, 10]^M$. As shown in Figure 4.4b, the coefficients obtained from our proxy teacher based method can achieve consistent improvements over the random coefficients. If we just randomly set the perturbation coefficients, the student performance can drop by up to 1.2%. Also, by comparing different perturbation orders, we find that the higher the perturbation order, the greater the performance differences. This is because in the higher-dimension space, it is harder for random search to get a set of appropriate perturbation coefficients, which makes the random PTLoss even worse than the standard KL loss. Conversely, equipped with the perturbation coefficients obtained via proxy teacher, PTLoss can significantly outperform the underlying KL loss.

4.6.4 Experiments on the CIFAR-100 dataset

Table 4.2: Results on CIFAR-100 dataset.

	MobileNetV2	ShuffleNetV2	ResNet18	GoogleNet	DenseNet121	ResNeXt29
Baseline	68.38	70.34	75.87	78.72	79.04	81.03
Tf-KD	70.14 ± 0.08	71.64± 0.17	76.60± 0.06	79.62± 0.43	79.54± 0.16	80.75± 0.13
PTLoss	**70.62 ± 0.16**	**71.97± 0.10**	**77.55 ± 0.06**	**80.22 ± 0.11**	**80.22± 0.11**	**81.83 ± 0.21**

To evaluate PTLoss on diverse tasks, we conduct this set of experiments on the CIFAR-100 dataset. We follow [114] to get the baseline results on the studied networks. Then we re-implement Tf-KD in [114] as we don't have access to the ground truth data during the distillation stage. Specifically, the Tf-KD implementation is modified from Tf-KD_self to be incorporated into our setting: we modified Eq.(7) in [114] as $L_{self} = D_{\text{KL}}(p^t_\tau, p_\tau)$, where p_τ and p^t_τ are the output probability of the student model and the pre-trained student model, reshaped by a temperature τ. For PTLoss, we simply add 1-order perturbation to p^t_τ, the ϵ is selected from $\{0.1, 0.2, 0.5\}$. From Table 4.2, we observe that PTLoss can still outperform those baselines, which shows the applicability of PTLoss on computer vision tasks.

4.7 Conclusions

In this study, we proposed a novel knowledge distillation loss PTLoss which implicitly shifts the teacher model output distribution to a high-fidelity one for student model training. We also established connections between PTLoss and other loss functions by demonstrating that PTLoss can subsume the others while providing more flexible adjustments to teacher models. We theoretically showed how the teacher model affects the student model risks and presented a principled method to systematically search perturbation coefficients. Extensive experiments on multiple tasks verified our proposed theory and validated the effectiveness of PTLoss.

While PTLoss enables better KD by creating a proxy teacher closer to the ground truth distribution, we focus on the single-teacher-single-student setting in this work. It is worth exploring how this approach can be extended to ensemble KD involving multiple teachers or students. Additionally, although the proposed coefficients selection method provides a principal way to determine the perturbation hyperparameters, it remains challenging to scale up the number of classes and the perturbation order. Future work could benefit from developing scalable methods for hyperparameter search, enabling rapid determination of perturbation coefficients even in high-dimensional spaces with numerous classes or high perturbation orders.

CHAPTER 5
ENHANCING INFERENCE EFFICIENCY VIA PROGRESSIVE KV CACHE COMPRESSION

5.1 Introduction

Autoregressive large language models (LLMs) such as GPT [94], PaLM [141], and LLaMA [142], built upon transformer architectures [143], have shown remarkable capabilities across a wide range of tasks. However, the attention mechanism underpinning those models poses significant challenges to the efficiency of their deployment, particularly the management of the Key-Value (KV) cache. The KV cache is originally designed to accelerate the generation process by storing intermediate attention KV vectors, thus avoiding recomputation of shared prefixes for each autoregressively generated token. Despite reducing computational overhead, the KV cache significantly increases memory footprints, as its size scales linearly with both sequence length and batch size. This drives the need for KV cache compression to enable cost-effective deployment of LLMs across various devices and platforms.

To address the overhead of the original attention mechanism, one prominent line of work aims to design more efficient attention variants, such as multi-query attention (MQA) [144] and group-query attention (GQA) [145], which inherently reduce the corresponding KV cache. Nevertheless, those techniques typically require upcycling existing models. Without proper training, their direct application often results in degraded performance [146, 145, 147], thereby making them unsuitable for deployment in resource-constrained environments. Recently, [148] design a multi-head latent attention (MLA) for efficient inference, utilizing low-rank key-value union compression to reduce KV cache. However, similar to MQA and GQA, MLA is also integrated during the model's training cycle, thus not directly applicable to pre-trained LLMs.

In contrast, another line of work focuses on KV cache compression at test time, primarily achieved by dropping tokens while leaving the backbone model intact. Several works design the token eviction policy based on accumulated attention scores[149, 150, 147], or heuristics such as special tokens or and relative distance between tokens [151] However, these methods either ignore the inter-layer dependency or require attention pattern analysis, and the resulting eviction policy can be task-specific.

In this paper, we propose to compress KV cache from an orthogonal perspective, *i.e.,* the KV weight matrices. As the KV weight matrices are typically characterized by low-rank properties, we perform a *low-rank approximation* to reduce their dimension and thus compress the resulting KV cache. Recognizing that compressed KV caches inevitably introduce information loss to subsequent layers, and that sensitivity to input changes varies across layers, we introduce a *progressive* compression strategy. This approach is grounded in the calculation of cumulative condition numbers for KV weight matrices across different layers, reflecting their sensitivity and guiding the compression strategy. Theoretically, we derive error bounds for both individual layer compression and error propagation through the network. These theoretical results reveal that errors introduced in earlier (shallower) layers are amplified more significantly than those in deeper layers, and informs our progressive compression strategy.

Our method is designed for straightforward implementation, requiring neither model profiling nor detailed inspection of the attention structure. It can be directly applied to pre-trained LLMs by extracting weight matrices and leveraging their inherent properties to swiftly determine optimal layer-wise compression. This approach offers a practical and efficient solution for enhancing LLM performance in memory-constrained deployment scenarios, without the need for model retraining or complex eviction strategy composition.

We evaluate our method on 8B, 13B, and 70B LLaMA models that built upon multi-query attention or group-query attention. Experiments across tasks such as commonsense reasoning, reading comprehension, text summarization, and mathematical reasoning, demon-

strate that our approach can reduce substantial GPU memory footprint while maintaining minimal impact on performance.

5.2 Related Works

5.2.1 Attention Mechanism

Attention mechanisms in Transformer models have evolved to enhance efficiency and effectiveness [143]. Multi-Query Attention (MQA)[144] reduces memory requirements during decoding, while Grouped-Query Attention (GQA) [145] balances efficiency and performance by sharing key and value heads among query groups. Recently, [148] introduced Multi-head Latent Attention (MLA), using low-rank key-value union compression to optimize inference. However, these approaches are typically integrated during model training, limiting their applicability to pre-trained LLMs. Parallel research efforts have targeted inference efficiency improvements. For example, [152] developed multi-dimensional partitioning techniques, and [153] optimized the Fusion-in-Decoder (FiD) approach [154] for more efficient inference. [155] introduces SplitFuse which leverages dynamic prompt and generation decomposition and unification to further improve continuous batching and system throughput. In this paper, we contribute to this line of research by improving inference efficiency through the compression of KV cache. Our approach leverages the low-rank property of the attention weight matrices, offering a plug-and-play method to reduce the memory footprint of LLMs during inference without requiring model retraining.

5.2.2 KV Cache Compression

As Large Language Models (LLMs) continue to grow in size and complexity, efficient management of their memory usage during inference has become a critical challenge. Early efforts to compress token hidden states [156, 157, 158] are limited to non-autoregressive models and require retraining, thus motivating research into pruning tokens in the KV cache of auto-regressive LLMs. For instance, [159] learns to compress prompts into a few

special tokens to reduce memory pressure during caching, but this token prediction requires model retraining and could be an expensive overhead during inference. Several methods design token eviction policies based on accumulated attention scores [149, 150, 147], or heuristics such as special tokens and relative distance between tokens [151]. However, these approaches often overlook inter-layer dependencies, potentially resulting in task-specific eviction policies that may not generalize well across different applications. In contrast to token-dropping methods, our study takes a different tack. We focus on compressing the KV cache from the perspective of weight matrix dimension reduction. Importantly, our progressive compression strategy carefully addresses the issue of error propagation across compressed layers, a consideration often ignored in previous methods.

A few studies have explored customized cache budgets across different layers in the context of token dropping, yet no definitive consensus has been reached on the most effective strategies. [160] suggest increasing compression intensity in higher layers based on the assumption that these layers contain less critical information. Conversely, [147] argue that significant tokens exhibit greater variability at higher layers, thus larger caches are required to reduce cache misses. While these approaches demonstrate understanding of layer-specific requirements, they depend heavily on task-specific attention patterns. Our approach diverges fundamentally by adopting an orthogonal perspective to compression, focusing on weight matrix dimension reduction rather than token eviction. This approach enables us to establish error propagation bounds across the network and to guide our progressive compression strategy effectively. It eliminates the need to analyze attention patterns for eviction policy design, simplifying implementation and enhancing general applicability across different LLMs.

Concurrently, [148] and [161] modify attention mechanisms to manage KV caches more efficiently during inference. While these methods align with our philosophy of altering attention dynamics, they require either pretraining adjustments or extensive model finetuning to accommodate the modified attention schemas, limiting their practicality in

deployed systems. In contrast, our method requires no such training or fine-tuning, offering a plug-and-play solution that seamlessly integrates with pre-trained models to deliver efficient compression without compromising the model's integrity or performance.

5.3 Preliminary: Attention Mechanism and KV Cache

Transformer-based language models use self-attention to weigh the importance of different tokens, thus allowing for the model to focus on different parts of the input sequence. Given an input $X \in \mathbb{R}^{N \times D}$, where N is the sequence length and D is the dimensionality of each token's embedding, we compute the Query (Q), Key (K), and Value (V) matrices by multiplying X with their respective weight matrices: $Q = XW_q, K = XW_k, V = XW_v$.

Then the attention mechanism is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{5.1}$$

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}^1, \ldots, \text{head}^h)W_o, \tag{5.2}$$

where

$$\text{head}^i = \text{Attention}(X(W_q^i)^T, X(W_k^i)^T, X(W_v^i)^T).^1 \tag{5.3}$$

Here, W_q^i, W_k^i, and W_v^i are the weight matrices for the i-th attention head, and W_o is the weight matrix for the output linear transformation.

In autoregressive transformers, the computation of attention scales quadratically (*i.e.,* $\mathcal{O}(N^2)$) with the sequence length N, as every token in the sequence computes interactions

[1]This formulation with transposed weight matrices aligns with the implementation found in the models examined in our study. Mathematically, this is equivalent to the standard formulation without transpose. The choice of which form to use depends on implementation details and computational optimizations.

with every other token. Such scaling is impractical for very large inputs or real-time applications, where speed and efficiency are crucial.

To address this computational bottleneck, KV caches store the results of previous computations of the KV matrices. When processing subsequent tokens, the model can retrieve keys and values from the cache rather than recomputing them, thereby reducing the number of operations to a linear scale with respect to the sequence length. This method trades off increased memory usage for a reduction in computational overhead. The size of KV cache per layer is defined as below:

$$C_{k,v} = b \times N \times h \times d, \tag{5.4}$$

where b is the batch size, N is the max sequence length in the batch, h is the number of K/V head and d is the head dimension. This linear relationship between cache size and sequence length, as well as batch size, underscores the critical need for efficient compression methods. As described, existing works that can reduce KV cache consumption either require expensive model training in upcycling stages or empirical token eviction policy design at test time. In the following section, we present a novel method for KV cache compression from the perspective of low-rank weight approximation.

5.4 Method

We structure this section as follows. In subsection 5.4.1, we detail the process of compressing the KV cache for a single layer using Singular Value Decomposition (SVD) on weight matrices. Next, subsection 5.4.2 introduces our progressive compression strategy, which determines adaptive compression dimensions for each layer. Finally, subsection 5.4.3 covers additional considerations for handling various attention mechanisms, and section H.1 addresses the implementation details specific to the rotary position embedding. Figure 5.1 presents an overview of our method, illustrating the low-rank approximation of the weight

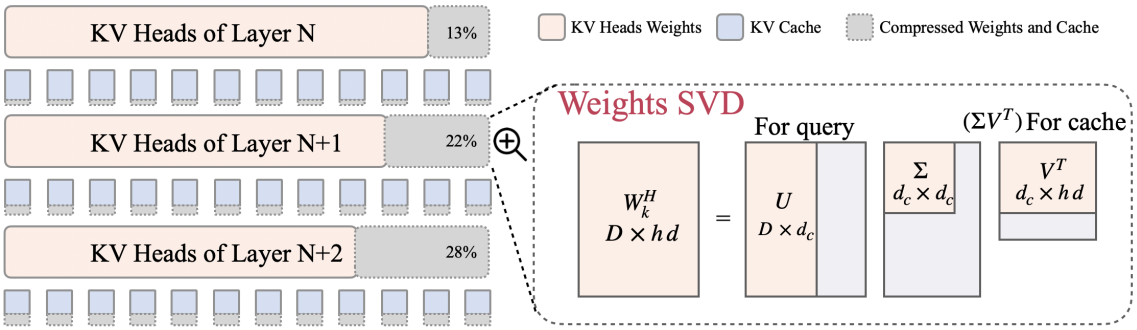


Figure 5.1: LoRC compresses KV-cache by decomposing the KV weight matrices in attention heads. The progressive compression strategy retains more dimension for KV weights in shallow layers and compresses the KV weights in deep layers more aggressively.

matrix and the progressive compression strategy across layers.

5.4.1 KV Cache Compression via Low-rank Approximation of Weight Matrices

Unlike previous approaches that focus on token-level eviction strategies or require model retraining, we propose a novel method that operates at the weight matrix level in the attention mechanism. This approach leverages the inherent low-rank properties of these matrices (as shown in Appendix section G.1), allowing for significant compression without the need for complex token selection algorithms or time-consuming model tuning. By applying a low-rank approximation to the weight matrices, we effectively reduce the dimensionality of the KV cache while preserving the essential information flow through the network.

Key Matrix Compression: Figure 5.1 presents how we implement SVD on the key weight matrices. Specifically, for the i-th head in the MHA attention, we decompose its key matrix $W_k^i \in \mathbb{R}^{D \times d}$ to:

$$\text{SVD}(W_k^i)_{D \times d} = U_{D \times d_c} \Sigma_{d_c \times d_c} V_{d_c \times d}^T = U_{D \times d_c} (\Sigma V^T)_{d_c \times d}. \tag{5.5}$$

For MHA, there are h attention heads, then the decomposition becomes:

$$\text{SVD}(W_k^H)_{D\times hd} = U_{D\times d_c}(\Sigma V^T)_{d_c\times hd} = U_{D\times d_c}\begin{bmatrix} (A^1)_{d_c\times d} & (A^2)_{d_c\times d} & \cdots & (A^h)_{d_c\times d} \end{bmatrix}, \tag{5.6}$$

where $(A^i)_{d_c\times d}$ is the i-th block in the matrix $(\Sigma V^T)_{d_c\times hd}$.

Now we have decomposed the key matrix W_k^i to the multiplication of $U_{D\times d_c}$ and $(\Sigma V^T)_{d_c\times hd}$. We will multiply X with $(\Sigma V^T)^T_{hd\times d_c}$ as the compressed key, which is stored in the KV cache. Through this implementation, we effectively update the size of key cache from hd to d_c, where d_c is smaller than hd, reducing the memory footprint while keeping the essential information intact.

For $U_{D\times d_c}$, we incorporate it to the query calculation by updating the original query matrix $W_q^H \in \mathbb{R}^{D\times hd}$ as follows:

$$W_{q'}^H = (W_q^H)_{D\times hd}U_{D\times d_c}. \tag{5.7}$$

Note that the embedding dimension D is equal to the product of the number of attention heads h and the dimension per head d, i.e., $D = hd$. Consequently, the updated query matrix $W_{q'}^H \in \mathbb{R}^{D\times d_c}$.

Value Matrix Compression: The decomposition for the value matrix follows a similar structure to that of the key matrix, with the difference that we integrate its left singular vectors to the output matrix W_o. Specifically, the value matrix is decomposed as:

$$\text{SVD}(W_v^H)_{D\times hd} = U_{D\times d_c}(\Sigma V^T)_{d_c\times hd} = U_{D\times d_c}\begin{bmatrix} (B^1)_{d_c\times d} & (B^2)_{d_c\times d} & \cdots & (B^h)_{d_c\times d} \end{bmatrix} \tag{5.8}$$

where $(B^i)_{d_c\times d}$ is the i-th block in the matrix $(\Sigma V^T)_{d_c\times hd}$. After multiplication with X, the dimension of the value cache shrinks from hd to d_c, thus reducing memory consumption.

In contrast to the key matrix operation, we incorporate $U_{D\times d_c}$ to the output matrix. To

achieve this, we update the output matrix $W_o \in \mathbb{R}^{D\times D}$ as follows:

$$W_{o'} = (U^\top)_{d_c\times D}(W_o)_{D\times D}, \tag{5.9}$$

resulting in an updated output matrix $W_{o'} \in \mathbb{R}^{d_c\times D}$.

Compression Ratio: The compression strategy effectively reduces the dimensions from $N \times d \times h$ for both keys and values to $N \times d_c$, ensuring data integrity and minimizing overhead. This results in a layer compression ratio $\rho = \frac{d_c}{h\times d}$, which quantifies the extent of the reduction.

5.4.2 Progressive Compression Strategy

Algorithm 3 LoRC Algorithm

Require: Pre-trained LLM with L layers

Initialize cumulative condition numbers $\tilde{\kappa}_l$
for $l = L$ **down to** 1 **do**
    Compute condition numbers $\kappa(W_k^l)$, $\kappa(W_v^l)$ and set $\tilde{\kappa}_l \leftarrow \prod_{j=l}^{L} \kappa(W_k^j) \cdot \kappa(W_v^j)$
end for
for $l = 1$ **to** L **do**
    Compute compression dimension d_c^l using Equation 5.13
    if $\tilde{\kappa}_l >$ threshold **then**
        Skip compression for layer l and **continue**
    end if
    Key Matrix Compression: Perform SVD: $W_k^l = U_k \Sigma_k V_k^T$
    Set $\tilde{W}_k^l \leftarrow U_k[:, : d_c^l](\Sigma_k V_k^T)[: d_c^l, :]$ and $W_{q'}^l \leftarrow W_q^l U_k[:, : d_c^l]$
    Value Matrix Compression: Perform SVD: $W_v^l = U_v \Sigma_v V_v^T$
    Set $\tilde{W}_v^l \leftarrow U_v[:, : d_c^l](\Sigma_v V_v^T)[: d_c^l, :]$ and $W_{o'}^l \leftarrow U_v[:, : d_c^l]^T W_o^l$
    Update KV cache size for layer l
end for

Having established low-rank approximation for compressing weight matrices, we now address its dynamic application across network layers. This approach is necessary due to

the varying sensitivity of different layers, which significantly affects overall model efficacy and efficiency.

To tackle this challenge, we propose a *progressive* compression strategy for our low-rank approximation of KV weight matrices. Our intuition is that the compressed shallow layers could lead to cascading errors that propagate and amplify through the network. Therefore, we measure the layer sensitivity by the condition numbers of KV matrices to determine *layer-wise* compression dimensions. This approach accounts for each layer's sensitivity to perturbations caused by previously compressed layers, ensuring output variations remain within acceptable ranges.

This progressive nature allows for more conservative compression in shallow layers and more aggressive compression in deeper layers, minimizing the risk of error accumulation throughout the network.

By carefully balancing compression across layers, we maintain model integrity while achieving significant memory savings.

Condition Number and Sensitivity Analysis To ensure that the change in the output $\mathbf{b}_l = \mathbf{A}_l\mathbf{x}_l$ remains within a specified range when the input $\mathbf{x}_l$ changes due to compression in previous layers, we need to consider the sensitivity of the output to such changes. Given a weight matrix $\mathbf{A}_l$, its condition number plays a crucial role in determining the allowable change in $\mathbf{x}_l$. The condition number $\kappa(\mathbf{A}_l)$ is defined as:

$$\kappa(\mathbf{A}_l) = |\mathbf{A}_l|_2 \cdot |\mathbf{A}_l^{-1}|_2 = \frac{\sigma_{\max}(\mathbf{A}_l)}{\sigma_{\min}(\mathbf{A}_l)}, \tag{5.10}$$

where $\sigma_{\max}(\mathbf{A}l)$ and $\sigma_{\min}(\mathbf{A}_l)$ are the largest and smallest singular values of $\mathbf{A}_l$, respectively. To keep the relative change in the output $\mathbf{b}_l$ within a tolerance ϵ, we utilize the standard definition of the condition number to relate it to the allowable relative change in the input $\mathbf{x}_l$:

$$\frac{|\Delta\mathbf{b}_l|_2}{|\mathbf{b}_l|_2} \leq \kappa(\mathbf{A}_l) \cdot \frac{|\Delta\mathbf{x}_l|_2}{|\mathbf{x}_l|_2} \leq \epsilon. \tag{5.11}$$

Solving for the allowable relative change in $\mathbf{x}_l$, we obtain: $\frac{|\Delta \mathbf{x}_l|_2}{|\mathbf{x}_l|_2} \leq \frac{\epsilon}{\kappa(\mathbf{A}_l)}$. This inequality indicates that the acceptable change in the input $\mathbf{x}_l$ is *inversely proportional* to the condition number $\kappa(\mathbf{A}_l)$ of the layer's weight matrix. Layers with higher condition numbers are more sensitive to input perturbations, requiring smaller changes in $\mathbf{x}_l$ to maintain the output within the desired range. Given the multi-layer structure of transformers, it is essential to consider not just the condition number of a single layer but the cumulative effect of condition numbers from all preceding layers. This cumulative measure gives a more holistic view of how perturbations might propagate and amplify as data passes through successive layers.

Cumulative Condition Number: To effectively manage this across the network, we calculate the cumulative condition number as an estimated layer sensitivity, which we then use to derive the compression dimension. For a model with L layers, we calculate the cumulative condition number for each layer l by multiplying the condition numbers of the current layer and all subsequent layers:

$$\tilde{\kappa}_l = \prod_{j=l}^{L} \kappa(W_k^j) \cdot \kappa(W_v^j), \tag{5.12}$$

where W_k^j and W_v^j denote the key and value weight matrices of the j-th layer, respectively. This cumulative condition number $\tilde{\kappa}_l$ reflects the total amplification of input perturbations from current layer to the final output layer, encompassing the effects of layers from l to L.

Compression Dimension: Based on the cumulative condition number, we then adjust the compression dimensions for each layer to balance the fidelity and compression rate. More sensitive layers (those with higher cumulative condition numbers) will have less aggressive compression to preserve information, whereas layers with lower sensitivity can be compressed more substantially without significantly affecting the overall network performance. We compute the compressed dimension d_c^l for each layer by scaling $\tilde{\kappa}_l$ using

the following function:

$$d_c^l = d_{\max} \times \left[1 - \left(\frac{\max_{i\in[1:L]} \log(\tilde{\kappa}_i) - \log(\tilde{\kappa}_l)}{\max_{i\in[1:L]} \log(\tilde{\kappa}_i) - \min_{i\in[1:L]} \log(\tilde{\kappa}_i)}\right) \times \left(1 - \frac{d_{\min}}{d_{\max}}\right)\right], \quad (5.13)$$

where $d_{\max}$ is the maximum allowable compressed dimension, and $d_{\min}$ is the minimum one. The logarithmic scale mitigates the effect of large variations in the cumulative condition numbers, providing a more balanced sensitivity metric across layers. This equation ensures that layers with higher sensitivity (larger $\tilde{\kappa}_l$) retain more dimensions (larger d_l), while less sensitive layers can be compressed more aggressively.

5.4.3 Multi-head Attention and Group-query Attention

The above derivation in subsection 5.4.1 holds for standard MHA, where the model dimension D equals to the multiplication of number of head and head dimension $h \times d$. For GQA, the number of KV heads is reduced as shown in Table G.1. To adapt such implementation, we can still follow the above procedure for cache compression. After fetching the key and value from cache, we just need to repeat them according to the number of the total attention heads.

5.5 Error Bounds for KV Cache Compression

In this section, we derive error bounds for our KV cache compression method, considering both individual layer errors and their propagation through a deep network. These theoretical results provide insights into how the matrix decomposition-based compression affects the network's performance and guide the progressive compression strategy to balance model efficiency and performance.

5.5.1 Error Bound for Key/Value Matrix Approximation

Theorem 2 *Let $W \in \mathbb{R}^{m\times n}$ be a weight matrix (either key or value), and let $\tilde{W} \in \mathbb{R}^{m\times n}$ be its rank-k approximation obtained via truncated singular value decomposition (SVD). For any input vector $x \in \mathbb{R}^n$, the error introduced by the approximation is bounded by:*

$$\|Wx - \tilde{W}x\|_2 \leq \sigma_{k+1}\|x\|_2, \tag{5.14}$$

where σ_{k+1} is the $(k+1)$-th singular value of W.

The proof is provided in subsection F.1.1. This theorem quantifies the error introduced at a single layer due to compressing the weight matrix. The bound indicates that the error is directly proportional to the $(k+1)$-th singular value of W and the norm of the input vector x. Larger singular values correspond to directions of significant variance in the data, so truncating smaller singular values (which represent less significant features) minimizes the error introduced by compression.

5.5.2 Single Layer Error Bound Including Nonlinearities

We now extend the analysis to include the effect of nonlinearities within a single layer. We derive an error bound that accounts for both the approximation of the weight matrix and the layer's nonlinear activation function. For simplicity, we analyze the error introduced by compressing each weight matrix (key or value) individually.

Theorem 3 *Consider a single layer applying a linear transformation W followed by a nonlinearity ϕ with Lipschitz constant L_ϕ. Let $\tilde{W}$ be the compressed version of W obtained via truncated SVD with rank k. For any input vector $x \in \mathbb{R}^n$, the error at the output of the layer is bounded by:*

$$\left|\phi(Wx) - \phi(\tilde{W}x)\right| \leq L_\phi \sigma_{k+1}\|x\|_2. \tag{5.15}$$

The proof is straightforward by using Theorem 1 and the Lipschitz property of ϕ, we present it as the base case in the proof of Theorem 3, which is detailed in subsection F.1.2.

This theorem shows that the error introduced by the compressed weight matrix propagates through the nonlinearity, scaled by the Lipschitz constant of the activation function. While considering both matrices simultaneously complicates the bounds due to their interactions within the attention mechanism, it is still feasible to derive combined error bounds because the attention mechanism allows us to mathematically bound these interactions. The total error due to simultaneous compression can be bounded by the sum of their individual approximation errors, scaled by a constant. However, for simplicity and clarity in the following derivation, we use the simplified version that considers each matrix individually.

5.5.3 Error Propagation Bound

Theorem 4 *Consider an L-layer network where each layer i applies a linear transformation W_i followed by a nonlinearity ϕ with Lipschitz constant L_ϕ. Let $\tilde{W}_i$ be the compressed version of W_i obtained via truncated SVD with rank k_i. The error at the output of the network is bounded by:*

$$\|x_L - \tilde{x}_L\|_2 \leq \sum_{i=1}^{L} \left(\sigma_{k_i+1}^{(i)} L_\phi^{L-i} \prod_{j=i+1}^{L} \|W_j\|_2 \right), \tag{5.16}$$

where x_L and $\tilde{x}_L$ are the outputs of the original and compressed networks, respectively; $\sigma_{k_i+1}^{(i)}$ is the (k_i+1)-th singular value of W_i; $\|W_j\|_2$ denotes the spectral norm of W_j; and L_ϕ is the Lipschitz constant of the activation function ϕ.

We detail the proof in subsection F.1.2. Until now, we have established an upper bound on the cumulative error at the network's output due to compression of weight matrices across multiple layers. It is important to note that the nonlinearities characterized by the Lipschitz constant L_ϕ represent a simplification. In practice, transformer models like LLaMA incorporate complex nonlinear components, so the exact error propagation may deviate

from this simplified bound due to intricate nonlinearities. Despite these complexities, the theorem still offers insights into how compression errors may accumulate in deep networks. Specifically, it reveals that errors introduced in earlier (shallower) layers are amplified more significantly than those in deeper layers because they pass through more subsequent transformations and nonlinearities.

This understanding supports our design of a progressive compression strategy, where we compress shallow layers less aggressively than deeper ones. By preserving more information in the early layers (i.e., retaining more singular values), we minimize the initial errors that could be significantly amplified throughout the network. This approach helps maintain overall model performance while still achieving substantial compression in deeper layers, where the impact on the final output is less pronounced due to reduced error amplification.

5.6 Experiment

5.6.1 Models

We conduct experiments using two attention mechanisms, Multi-Head Attention (MHA) [143] and Graph Query Attention (GQA) [145], across three models: LLaMA-2-13B, LLaMA-3-Instruct-8B, and LLaMA-3-Instruct-70B. The LLaMA-2 family incorporates the MHA mechanism, while the LLaMA-3 family is based on the GQA framework. We list the model specifications in Table Table G.1. Note that for the models based on MHA, the number of KV heads is equal to the number of attention heads, so the weight matrices of KV are square matrices. The models based on GQA use an intermediate number of key-value heads to group the query heads, with an adjustment on the shape of KV weight matrices.

5.6.2 Implementation Details

In practice, we set thresholds to exclude compression on layers with high cumulative condition numbers: 30 for LLaMA-3-Instruct-8B, and 90 for LLaMA-2-13B and LLaMA-3-Instruct-70B. The d_{max} equals to the original head dimension, while d_{min} varies based

on the target compression ratio. For baseline methods, we have the same refrained layers while applying the uniform compression ratios across compressed layers instead of using a progressive compression strategy.

5.6.3 Dataset

We follow [142] to evaluate our methods on the following tasks: **BoolQ** [136] for reading comprehension, **XSum** [162] for text summarization. **Openbook QA** [163] for commonsense reasoning, and **GSM8K** [164] for mathematical reasoning. We use ROUGE score [165] as the evaluation metric for XSum and accuracy for the other tasks. We report 2-shot results for LLaMA-2 models on BoolQ, and 0-shot results for other settings.

5.6.4 Main Results

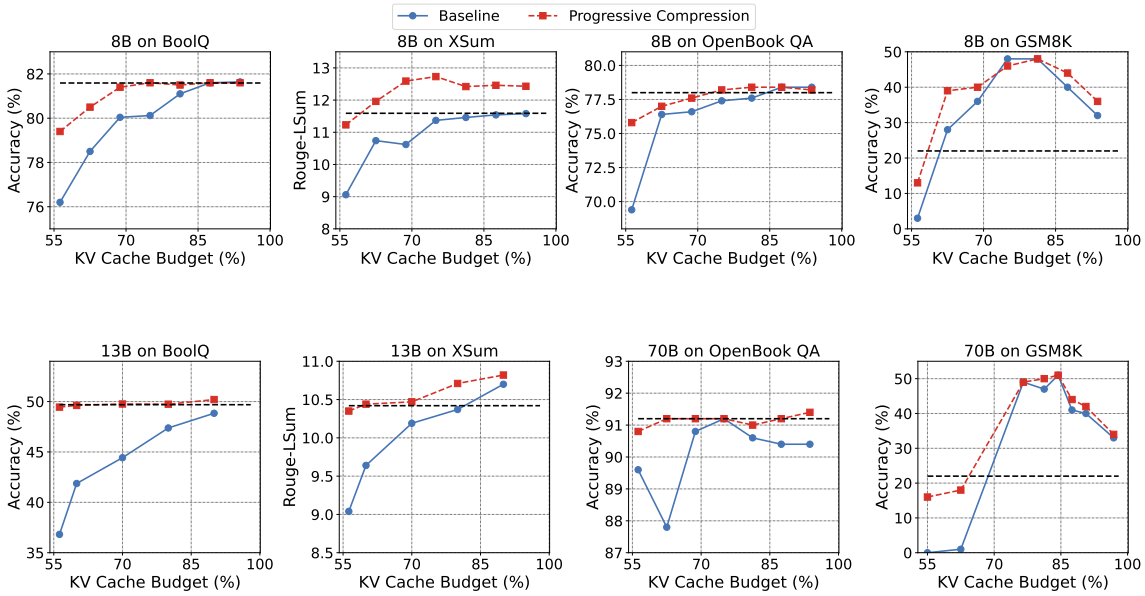


Figure 5.2: Performance of KV cache compression on LLaMA models. LoRC compresses the KV weights with a progressive strategy, while the baselines compress each layer with the same ratio. The horizontal dashed line indicates the performance with a full-cache model.

Figure 5.2 presents our main results on four datasets with different KV cache budgets. Compared to the full-cache model, LoRC achieves on-par performance with a significant

compression ratio, and the performance degradation is still nearly negligible with a 60% compression ratio on most datasets. When slightly compressed, LoRC could even enhance model performance in some cases. Note that our method requires no model training or model profiling, the only efforts are SVD on weight matrices which requires minimal computational cost compared to the LLM inference. Such plug-and-play merits make our method easily integrable in resource-constrained environments, enabling efficient model deployment with limited KV cache budgets.

In Figure 5.2, one interesting observation is that in some cases the model with a compressed KV cache leads to better performance. Particularly, on the GSM8K dataset, performing KV cache compression leads to more than 10% performance improvement. This phenomenon aligns with findings reported in the literature [151]. Also, similar effects have been documented in the context of improving reasoning by applying low-rank decomposition on the MLP layers [166]. We believe this phenomenon demonstrates the feasibility of conducting task-specific profiling for better performance, or adapting our proposed method in model finetuning.

5.6.5 Single Layer Profiling

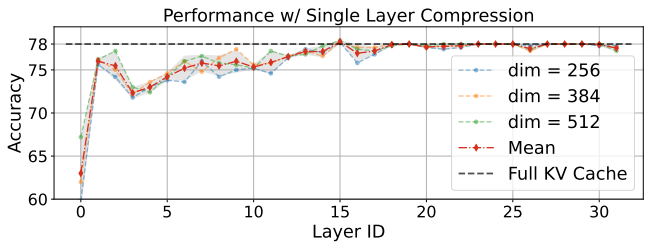


Figure 5.3: Single-layer compression results. This experiment uses LLaMA-3-Instruct-8B on the OpenBookQA dataset.

To investigate the impact of compression at different layers, we conduct experiments on single-layer compression as shown in Fig. Figure 5.3. We use LLaMA-3-Instruct-8B on OpenBook QA for this experiment. The original dimension of the KV head is 1024, and we select compression dimensions from $[256, 384, 512]$ to compress each single layer while keeping all other layers untouched.

Figure 5.3 shows clear layer-specific variability, indicating that some layers are more

susceptible to compression than others, particularly in the shallow layers. It is observed that the deep layers (*i.e.,* layers 15–31 of the 32-layer LLaMA-3-Instruct 8B model), despite the reduction in dimensions, maintain performance closely approaching the full KV Cache baseline. This suggests that these layers can sustain robust performance even when subjected to significant parameter reduction. This finding supports our progressive compression strategy for optimizing model efficiency without significantly compromising the model's effectiveness.

5.6.6 Curse of shallow layers

Table 5.1: Performance comparison between compression on shallow layers and deep layers on OpenBookQA. For our progressive compression strategy, we report the performance at the 60% overall compression ratio. For layer-0 compression and shallow blocks compression, we use a 50% layer compression ratio within the chosen strategy. Hence, the overall compression ratio is 98.44% for the layer-0 compression, and 93.75% for the shallow blocks compression.

Model	Baseline	Ours	Layer 0	Shallow Blocks (1/8)
LLaMA-2-13b	76.6	77.4 (↑ 0.8)	77.2 (↑ 0.6)	74.8 (↓ 1.8)
LLaMA-3-Instruct-8b	78.0	77.4 (↓ 0.6)	67.2 (↓ 10.8)	61.4 (↓ 16.6)
LLaMA-3-Instruct-70b	91.2	91.2 (↑ 0.0)	84.2 (↓ 7.0)	23.2 (↓ 68.0)

To validate the intuition of the progressive compression strategy that the noise caused by shallow compressed layers will be amplified more after propagation, we compare it to compressing the first layer and the shallow blocks (i.e., the first 1/8 layers in a model) on 3 LLaMA models.

Table 5.1 shows how the compressed shallow layers impact the model performance, taking the baseline full-cache model and our method as reference. The results indicate that compressing only the first layer can lead to a performance decline, with reductions ranging from minimal to moderate. For instance, the LLaMA-3-70B gives a 7.0% decrease, while the LLaMA-3-Instruct-8b shows a more substantial drop of 10.8%. When compressing the shallow blocks, the impact is more pronounced. The LLaMA-3-Instruct-8B suffers a 16.6% reduction. Notably, the LLaMA-3-Instruct-70b model shows a drastic 68.0% decline,

highlighting a significant sensitivity to shallow layer compression.

These findings underscore the importance of careful layer selection in compression strategies and validate the effectiveness of our progressive compression method, as the choice of layer to compress can have a substantial impact on model performance, particularly in larger or more complex models.

5.6.7 Memory footprint reduction analysis

Table 5.2: Summary of Model Sizes, KV cache usage and performance drop. Experiments were conducted with a batch size of 64 and a sequence length of 2048 for all models.

Model	KV Cache					Average Performance Drop
	Full	dim	dim_c	Ours	Compression Ratio	
LLaMA-2-13B	50G	5120	2048	27.5G	55%	0.47%
LLaMA-3-8B	8G	1024	512	4.8G	60%	0.92%
LLaMA-3-70B	20G	1024	512	11G	55%	0.22%

We report the memory footprint reduction in Table 5.2. By controlling the performance drop averaged on the four tasks less than 1%, we can achieve a considerable compression ratio from 55%-60%. For the LLaMA-3 models in which the GQA has already been employed to save the KV cache, we further achieve a significant compression ratio. Note that we have excluded the GSM8k results for the performance drop calculation for a fair comparison.

5.7 Conclusions

In conclusion, we proposed LoRC, a novel approach to KV cache compression that capitalizes on the inherent low-rank properties of weight matrices. Our method employs a progressive layer-wise compression strategy, implementing a post-hoc low-rank approximation to circumvent the complexities and limitations associated with token-level eviction strategies and model retraining. Moreover, we provide theoretical analysis, deriving error bounds for layer compression and error propagation in deep networks, supporting our design

of progressive compression strategy. This theoretically grounded and universally applicable approach preserves model integrity and performance across diverse tasks, attention mechanisms, and model scales. Our comprehensive experimental results demonstrate that LoRC significantly reduces GPU memory requirements while minimally impacting performance. This approach offers a robust and efficient solution of KV cache compression, without requiring attention pattern analysis or model tuning.

Appendices

APPENDIX A

IMPLEMENTATION DETAILS OF PRBOOST

A.1 Dataset Details

Weak sources For each dataset above, we have an existing weak source that uses labeling rules to generate weakly labeled data.

1. *TACRED*: We use the rules in [10] for the relation extraction task. Their rules are in the form of relation phrases, which include the entity pair and a keyword.

2. *DBPedia*: We use the keywords provided in [8] as the labeling rules. Such keywords are indicative to the categories, where the words for the same category have close semantics.

3. *AGNews, ChemProt*: We use the rules in [20] as the labeling fucntions. They also extract lexical patterns for weak supervision.

Table A.1: More rule examples on the text classification dataset AG News and the relation extraction dataset TACRED.

Rule	**Label**
If `[Mask]` prediction is in {*Economic, Deal, Business, Market*}	Business
If `[Mask]` prediction is in {*Microsoft, Tech, Software*}	Sci/Tech
If `[Mask]` prediction is in {*African, Global, World*}	World
If `[Mask]` prediction is in {*NFL, Sports, Team, Football*}	Sports
If entity pair == (Organization, Organization) and `[Mask]` prediction is in {*formerly, called, aka*}	org:alternate_names
If entity pair == (Person, Organization) and `[Mask]` prediction is in {*founded, established, started*}	org:founded_by
If entity pair == (Person, Title) and `[Mask]` prediction is in {*president, head, chairman, director*}	org:top_members
If entity pair == (Person, City) and `[Mask]` prediction is in {*moved to, lived in, grew in*}	per:city_of_residence

Table A.2: Dataset statistics.

Dataset	**Task**	**Domain**	**# Class**	**# Train**	**# Test**
TACRED	Relation Extraction	Web Text	41	68,124	15,509
DBPedia	Ontology Classification	Wikipedia Text	14	560,000	70,000
Chemprot	Chemical-protein Interaction Prediction	Biology	10	5,400	1,400
AG News	News Topic Classification	News	4	120,000	7,600

A.2 Hyper-parameters

We show the hyper-parameter configuration in Table A.3. We search the batch size in $\{8, 16, 32, 64, 128\}$, AND the coefficient α between $[0, 1]$ with an interval of 0.25. For the optimizer, we use AdamW [167] and choose learning rate from $\{5\times10^{-6}, 1\times10^{-5}, 2\times10^{-5}\}$. We keep the number of iterations as 10 for all the tasks and show the top-10 candidate rules to solicit human feedback. ChemProt is a special case where we present the top-20 candidate rules, because this task is more domain-specific than the others, and the involved human annotators have no relevant domain background.

Table A.3: Hyper-parameter configurations.

Hyper-parameter	**TACRED**	**DBpedia**	**ChemProt**	**AG News**
Maximum Tokens	128	256	512	128
Batch Size	32	32	8	32
Learning Rate	2×10^{-5}	10^{-5}	10^{-5}	10^{-5}
Dropout Rate	0.2	0.1	0.1	0.1
# Iterations	10	10	10	10
α	0.5	0.25	0.5	0.25
k	10	10	20	10

A.3 Computational Setting

We test our code on the System *Ubuntu 18.04.4 LTS* with CPU: *Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz* and *GPU: NVIDIA GeForce RTX 2080*. We implement our method using Python 3.6 and PyTorch 1.2 [168].

A.4 Model Ensemble

In practice, we keep α_t for each weak model as same during the model ensemble. Equation 2.11 weights each weak model m_t by a computed coefficient α_t. Intuitively, the weak model m_t with higher α_t impacts the ensemble results more. This paradigm is proved

to be effective under fully-supervised settings, but we found it is not directly applicable in WSL. Since we initialize a model m_0 on the given weak source and it can achieve a relatively strong performance (much better than random guess), *i.e.*, the error rate err_0 is low. It makes a high α_0 based on Equation 2.2, so the initialized model will dominate the following prediction, thus limiting the effectiveness of the model ensemble. Therefore, we assign the same weight to each weak model but still follow the design of identifying large-error instances. This is reasonable as the weight w_i computed by Equation 2.1 still reflects the model weakness and can guide the rule proposal. By discovering rules based on the large-error instances, we iteratively complement the feature regimes through the model training on rule-matched data and strengthen the ensemble model.

APPENDIX B

ADDITIONAL RESULTS OF PRBOOST

B.1 Interactive Baselines and User Study

For interactive learning, we include three baseline methods: the interactive weak supervision (IWS) framework [18], the most recent active learning (AL) method CAL [27], and an entropy-based AL method. Our objectives are: (1) to compare the annotation costs associated with rule-level and instance-level annotations; and (2) to evaluate model performance under identical annotation budgets.

Since IWS was originally designed for binary classification tasks, we modify its implementation for multi-class tasks by integrating predictions from multiple binary classifiers. Specifically, we obtain predicted probabilities across all categories from each binary classifier and then select the category with the highest probability as the final prediction. This approach becomes computationally expensive when the number of categories increases, as training multiple classifiers requires significant time. Therefore, we evaluate IWS only on the AG News dataset, which contains four categories.

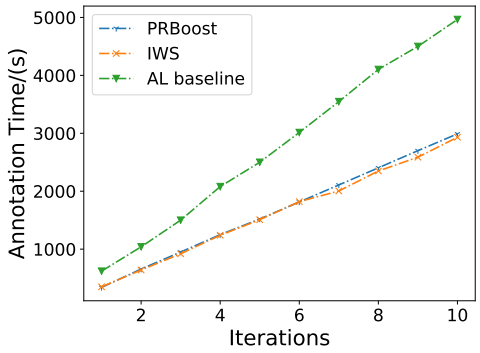


Figure B.1: Annotation cost of interactive methods measured by annotation time on AGNews. Both and IWS use rule-level annotation, while AL baselines use instance-level annotation.

To comprehensively assess annotation costs, we conducted a user study involving three

human annotators participating in a 10-iteration annotation experiment. In each iteration, annotators reviewed 100 candidate rules, providing binary decisions regarding their validity. We recorded the annotation time for each iteration and calculated the average annotation duration. Results comparing annotation times of interactive methods are presented in subsection 2.5.3 and detailed further in Figure Figure B.1. Both our proposed method and IWS utilize rule-level annotations, whereas the AL baselines employ instance-level annotations.

Additionally, we measured inter-annotator agreement for rule-level annotations using Fleiss' kappa (κ), defined as:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}, \tag{B.1}$$

where $\bar{P}$ represents the observed agreement across annotators, and $\bar{P}_e$ denotes the expected agreement calculated based on category assignments. As reported in subsection 2.5.3, the annotators demonstrated substantial agreement in rule-level annotations.

Finally, we generated candidate rules from publicly available pre-trained language models (PLMs) and public datasets. We do not anticipate that the rule-level annotation process will amplify existing biases in the original data or introduce ethical concerns or negative social impacts.

APPENDIX C

IMPLEMENTATION DETAILS OF PTLOSS

C.0.1 Dataset Statistics

We list the dataset statistics in Table C.1.

C.0.2 Hyper-parameters

We list the search range of hyperparamters in Table C.2. The search for batch size and learning rate is applied to all the methods. And for each baseline, we search for the best baseline-specific hyper-parameters.

C.0.3 Search Time of Perturbation Coefficients

For each perturbation order, we randomly sample 100 coefficient sets from $[-1, 10]$ and find the best set that has the lowest risk deviation gap according to Equation 4.20 with 1000 validation examples. The whole process takes less than two minutes on CPU with 64G memory.

Table C.1: Dataset Statistics

Dataset	Task	Train	Distillation	Dev	Test
CoLA	Linguistic Acceptability	8.5k	8.5k	3k	1k
MNLI	Natural Language Inference	58.9k	314k	19.6k	9.8k
MRPC	Paraphrase Similarity Matching	3.7k	3.7k	1k	1.7k
RTE	Textual Entailment Inference	2.5k	2.5k	0.8k	3k
SST-2	Sentiment Analysis	6.7k	53.8k	6.7k	872
BoolQ	Boolean Question Answering	2.5k	5.9k	1k	3.2k

Table C.2: The search range of hyper-parameters.

Hyper-parameter	**Search Range**
Learning Rate	$\{2, 3, 5\} \times 10^{-5}$
Batch Size	$\{8, 16, 32, 64, 128, 256\}$
Temperature T	$\{0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10\}$
Label Smoothing δ	$\{0.02, 0.05, 0.1, 0.15, 0.2\}$
Focal Loss τ	$\{0.1, 0.2, 0.5, 1, 2.0, 5.0\}$
Random $\epsilon_{c,m}$	$[-1, 10]$

APPENDIX D

PROOFS OF PTLOSS

Theorem 1 states that given a teacher model $\mathbf{p}^t$, an unlabeled distillation dataset $\mathcal{D}_u$ with an unknown true distribution $\mathbf{p}^*$, we have for any probability predictor $\mathbf{p} : \mathcal{X} \to \mathbb{R}^C$:

$$\begin{aligned}\mathbb{E}\left[(\tilde{R}_{KL}(\mathbf{p}; \mathbf{p}^t, \mathcal{D}_u) - R(\mathbf{p}))^2\right] \leq \frac{2}{N_u} \cdot \mathbb{V}\left[\mathbf{p}^t(x)^T \log(\mathbf{p}(x))\right] + \\ \mathcal{O}\left(\left(\mathbb{E}_x\left[\|\mathbf{p}^t(x) - \mathbf{p}^*(x)\|_2\right]\right)^2 + \right. \\ \left. \mathbb{E}_x\left[\left(\mathbf{p}^t(x)^T \log \mathbf{p}^t(x)\right)^2\right]\right),\end{aligned}$$

where $\mathbb{V}[\cdot]$ denotes the variance of a random variable.

We first rewrite the population risk $R(\mathbf{p})$ with cross-entropy loss l_{CE} plugged in as follow:

$$\begin{aligned}R(\mathbf{p}) &= \mathbb{E}_{(x,y)}[\ell(y, \mathbf{p}(x))] \\ &= \mathbb{E}_x[\mathbb{E}_{y|x}[\ell(y, \mathbf{p}(x))]] \\ &= \mathbb{E}_x[\mathbf{p}^*(x)^T(-\log(\mathbf{p}(x)))].\end{aligned} \tag{D.1}$$

Then, we write out the distillation empirical distillation risk defined in Equation 4.3 and have:

$$\begin{aligned}\tilde{R}_{KL}(\mathbf{p}; \mathbf{p}^t, \mathcal{D}_u) - R(\mathbf{p}) = \frac{1}{N_u}\sum_{n=1}^{N_u} \mathbf{p}^t(x_n)^T(-\log(\mathbf{p}(x_n))) \\ + \frac{1}{N_u}\sum_{n=1}^{N_u} \mathbf{p}^t(x_n)^T \log(\mathbf{p}^t(x_n)) - \mathbb{E}_x[\mathbf{p}^*(x)^T(-\log(\mathbf{p}(x)))].\end{aligned} \tag{D.2}$$

We let

$$\Delta \doteq \frac{1}{N_u}\sum_{n=1}^{N_u} \mathbf{p}^t(x_n)^T(-\log(\mathbf{p}(x_n))) - \mathbb{E}_x[\mathbf{p}^*(x)^T(-\log(\mathbf{p}(x)))],$$

and

$$H \doteq \frac{1}{N_u}\sum_{n=1}^{N_u} \mathbf{p}^t(x_n)^T \log(\mathbf{p}^t(x_n)),$$

then

$$\begin{aligned}\mathbb{E}\left[(\tilde{R}_{KL}(\mathbf{p};\mathbf{p}^t,\mathcal{D}_u)-R(\mathbf{p}))^2\right] &= \mathbb{E}\left[(\Delta+H)^2\right] \\ &\leq 2\mathbb{E}\left[\Delta^2\right]+2\mathbb{E}\left[H^2\right] \\ &= 2\mathbb{V}\left[\Delta\right]+2\mathbb{E}\left[\Delta\right]^2+2\mathbb{E}\left[H^2\right]\end{aligned} \tag{D.3}$$

where the second line is by the inequality $(a+b)^2 \leq 2a^2+2b^2$ and the linearity of expectation, and the third line is by $\mathbb{E}\left[\Delta^2\right] = \mathbb{V}\left[\Delta\right]+\mathbb{E}\left[\Delta\right]^2$. Observe that

$$\begin{aligned}\mathbb{E}\left[\Delta\right] &= \mathbb{E}_x\left[(\mathbf{p}^t(x)-\mathbf{p}^*(x))^T(-\log(\mathbf{p}^t(x_n)))\right] \\ &\leq \mathbb{E}_x\left[\|\mathbf{p}^t(x)-\mathbf{p}^*(x)\|_2\cdot\|\log(\mathbf{p}^t(x_n))\|_2\right] \\ &\leq \mathbb{E}_x\left[\|\mathbf{p}^t(x)-\mathbf{p}^*(x)\|_2\cdot c_1\cdot\|\log(\mathbf{p}^t(x_n))\|_\infty\right] \\ &\leq c_2\mathbb{E}_x\left[\|\mathbf{p}^t(x)-\mathbf{p}^*(x)\|_2\right],\end{aligned} \tag{D.4}$$

where the second line is by the Cauchy-Schwartz inequality, the third line is by the equivalence of norms with a constant c_1, and the last line is by the boundedness of the log loss term[1].

Furthermore, we notice the $R(\mathbf{p})$ term in the above Δ is a constant and thus have:

$$\begin{aligned}\mathbb{V}\left[\Delta\right] = \mathbb{V}\left[\tilde{R}_{KL}(\mathbf{p};\mathbf{p}^t,\mathcal{D}_u)\right] &= \frac{1}{N_u}\cdot\mathbb{V}\left[\mathbf{p}^t(x)^T(-\log(\mathbf{p}(x)))\right] \\ &= \frac{1}{N_u}\cdot\mathbb{V}\left[\mathbf{p}^t(x)^T\log(\mathbf{p}(x))\right],\end{aligned} \tag{D.5}$$

where the last equation comes from $\mathbb{V}[aX+b] = a^2\mathbb{V}[X]$.

Finally, we plug in Eqs (Equation D.4)(Equation D.5) and the definition of H into Eq. (Equation D.3) and complete the proof.

[1]This is a common assumption defined in previous literature such as (Boucheron *et al.* [169], Theorem 4.1; Menon *et al.* [105], Proposition 2) and can be achieved easily in practice with regularization techniques.

APPENDIX E

CONNECTIONS BETWEEN PTLOSS AND OTHER PERTURBATION METHODS

In subsection 4.5.1, we have presented the connection between PTLoss and other perturbation methods. The connection between PTLoss and the standard KL loss is direct, and it has been clearly presented how PTLoss subsumes the focal loss. Now we detail the derivation of the connection between PTLoss and temperature scaling and label smoothing here.

Temperature Scaling. We compare PTLoss with temperature scaling and claim that PTLoss subsumes it with appropriate approximation. As described in [104], the logits are adjusted by a temperature to control sharpness or smoothness of the probability distribution.

$$\mathbf{p}_{0,\tau} = \frac{exp(z_0/\tau)}{exp(z_0/\tau) + exp(z_1/\tau)}, \tag{E.1}$$

where τ is the temperature and z_c is the logits. Here we use binary classification (*i.e.,* $c = 0, 1$) for the derivation simplicity without loss of generality. Denote the probability without temperature scaling as $\mathbf{p}_0$, we have

$$\begin{aligned} \frac{\mathbf{p}_{0,\tau}}{\mathbf{p}_0} &= \frac{exp(z_0/\tau)}{exp(z_0/\tau) + exp(z_1/\tau)} \times \frac{exp(z_0) + exp(z_1)}{exp(z_0)} \\ &= \frac{1 + exp(z_1 - z_0)}{1 + exp(\frac{z_1 - z_0}{\tau})}. \end{aligned} \tag{E.2}$$

In practice, we have

$$\frac{1 + exp(z_1 - z_0)}{1 + exp(\frac{z_1 - z_0}{\tau})} \approx 1 \quad \text{or} \quad \frac{exp(z_1 - z_0)}{exp(\frac{z_1 - z_0}{\tau})} \tag{E.3}$$

because $|z_1 - z_0| \gg 0$. Then we have

$$\mathbf{p}_{0,\tau} \approx \mathbf{p}_0 \quad \text{or} \quad \frac{exp(z_1 - z_0)}{exp(\frac{z_1 - z_0}{\tau})}\mathbf{p}_0. \tag{E.4}$$

For the first case where $\mathbf{p}_{0,\tau} \approx \mathbf{p}_0$, we omit the discussion as it aligns with the standard KL loss. For the second case, we proceed to draw its connection with PTLoss as follows. We denote $\mathbf{p}^t_{\tau,c}$, $\mathbf{p}^s_{\tau,c}$ as the teacher, student probability scaled by temperature τ. Incorporating temperature scaling, the KL loss can be formulated as:

$$\ell^{temp}_{KL}\left(\mathbf{p}^t_\tau, \mathbf{p}^s_\tau\right) = -\mathbb{H}\left(\mathbf{p}^t_\tau\right) + \sum_{c\in[C]} \mathbf{p}^t_{\tau,c} \cdot (-\log \mathbf{p}^s_{\tau,c}), \tag{E.5}$$

Substituting $\mathbf{p}^t_{\tau,c}$ and $\mathbf{p}^s_{\tau,c}$ in Equation E.5 using Equation E.4, we obtain

$$\ell^{temp}_{KL}\left(\mathbf{p}^t_\tau, \mathbf{p}^s_\tau\right) = -\mathbb{H}\left(\mathbf{p}^t_\tau\right) + \sum_{c\in[C]} a \cdot \mathbf{p}^t_c \cdot [-\log(b \cdot \mathbf{p}^s_c)], \tag{E.6}$$

where $a = exp(z^t_1 - z^t_0)/exp(\frac{z^t_1 - z^t_0}{\tau})$, and $b = exp(z^s_1 - z^s_0)/exp(\frac{z^s_1 - z^s_0}{\tau})$.

Comparing above Equation E.6 with Equation 4.5, we can set

$$\sum_{m=1}^{\infty}(\frac{1}{m} + \epsilon_{c,m}) \cdot (1 - \mathbf{p}^s_c)^m = a[\sum_{m=1}^{\infty} \frac{1}{m} \cdot (1 - \mathbf{p}^s_c)^m + \sum_{m=1}^{\infty} \frac{1}{m} \cdot (1 - b)^m]. \tag{E.7}$$

It leads to

$$\epsilon_{c,m} = \frac{a}{m} \cdot (1 + (\frac{1 - b}{1 - \mathbf{p}^s_c})^m) - \frac{1}{m}, \tag{E.8}$$

which indicates PTLoss can encompass the temperature-scaled distillation loss by setting a group of appropriate perturbation coefficients.

Label Smoothing. We compare PTLoss with the label smoothing method and claim that label smoothing proposed in [110] is a special case of PTLoss. According to the

implementation in [110], we can smooth the teacher labels in KD by

$$\mathbf{p}_c^{t_{ls}} = (1-\delta)\mathbf{p}_c^t + \delta/2, \tag{E.9}$$

with a smoothing parameter δ. Starting from Equation 4.5, we can replace the term $\mathbf{p}_c^t$ by its smooth version $\mathbf{p}_c^{t_{ls}}$. Then the original Equation 4.5 with label smoothing becomes:

$$\ell_{KL}^{ls}\left(\mathbf{p}^t, \mathbf{p}^s\right) = -\mathbb{H}\left(\mathbf{p}^{t_{ls}}\right) + \sum_{c\in[C]} \mathbf{p}_c^{t_{ls}} \cdot (-\log \mathbf{p}_c^s) \tag{E.10}$$

For the entropy of the teacher output, the smooth version $\mathbb{H}\left(\mathbf{p}^{t_{ls}}\right)$ is different from the original $\mathbb{H}\left(\mathbf{p}^t\right)$ with only a constant C, which can be ignored when optimizing the loss function. We introduce $\Delta\mathbf{p}_c^t = \delta/2 - \delta\mathbf{p}_c^t$ and replace all the $\mathbf{p}_c^{t_{ls}}$ in Equation E.10 by $\mathbf{p}_c^{t_{ls}} = \mathbf{p}_c^t + \Delta\mathbf{p}_c^t$, then we get:

$$\ell_{KL}^{ls}\left(\mathbf{p}^t, \mathbf{p}^s\right) = -\mathbb{H}\left(\mathbf{p}^{t_{ls}}\right) + \sum_{c\in[C]} (\mathbf{p}_c^t + \Delta\mathbf{p}_c^t) \cdot (-\log \mathbf{p}_c^s) \tag{E.11}$$

Similarly, we let $\ell_{KL}^{ls}\left(\mathbf{p}^t, \mathbf{p}^s\right) = \ell_{PT}\left(\mathbf{p}^t, \mathbf{p}^s\right)$, it yields

$$\sum_{c\in[C]} (\mathbf{p}_c^t + \Delta\mathbf{p}_c^t) \sum_{m=1}^{\infty} \frac{1}{m} \cdot (1-\mathbf{p}_c^s)^m = \sum_{c\in[C]} \mathbf{p}_c^t \sum_{m=1}^{\infty} (\frac{1}{m} + \epsilon_{c,m}) \cdot (1-\mathbf{p}_c^s)^m. \tag{E.12}$$

We obtain

$$\epsilon_{c,m} = \frac{\Delta\mathbf{p}_c^t}{m\mathbf{p}_c^t} \xrightarrow{\Delta\mathbf{p}_c^t = \delta/2 - \delta\mathbf{p}_c^t} \epsilon_{c,m} = \frac{\delta}{m}(\frac{1}{2\mathbf{p}_c^t} - 1). \tag{E.13}$$

In summary, the connection between the two losses can be expressed through a specific $\epsilon_{c,m}$, which depends on the smoothing parameter δ. This derivation highlights that PTLoss generalizes the label smoothing method and provides a more flexible framework that encompasses the effects of label smoothing.

APPENDIX F

PROOFS OF LORC

F.1 Detailed Proofs

F.1.1 Proof of Theorem 2

The proof of Theorem 2 is presented here for completeness.

Proof.

Let $W = U\Sigma V^\top$ be the full SVD of W, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$.

The rank-k approximation $\tilde{W}$ is given by:

$$\tilde{W} = U_k \Sigma_k V_k^\top,$$

where U_k, Σ_k, and V_k are truncated versions of U, Σ, and V, respectively, keeping only the first k singular values and corresponding vectors.

We have:

$$\begin{aligned}
\|Wx - \tilde{W}x\|_2 &= \|(W - \tilde{W})x\|_2 \\
&= \|U(\Sigma - \Sigma_k)V^\top x\|_2 \\
&= \|(\Sigma - \Sigma_k)V^\top x\|_2, \quad \text{since } U \text{ is orthogonal} \\
&= \|\,\mathrm{diag}(0, \ldots, 0, \sigma_{k+1}, \ldots, \sigma_n)V^\top x\|_2 \\
&\leq \sigma_{k+1}\|V^\top x\|_2 \\
&= \sigma_{k+1}\|x\|_2, \quad \text{since } V \text{ is orthogonal.}
\end{aligned}$$

□

F.1.2 Proof of Theorem 4

We present the proof of Theorem 4, including an adjustment for activation functions with Lipschitz constant L_ϕ.

Proof.

Let x_i and $\tilde{x}_i$ denote the outputs of the i-th layer in the original and compressed networks, respectively. We prove by induction that:

$$\|x_i - \tilde{x}_i\|_2 \leq \sum_{s=1}^{i} \left(\sigma_{k_s+1}^{(s)} L_\phi^{i-s} \prod_{j=s+1}^{i} \|W_j\|_2 \right). \tag{F.1}$$

Base Case ($i = 1$).

Using Theorem 2 and the Lipschitz property of ϕ:

$$\begin{aligned} \|x_1 - \tilde{x}_1\|_2 &= \|\phi(W_1 x_0) - \phi(\tilde{W}_1 x_0)\|_2 \\ &\leq L_\phi \|W_1 x_0 - \tilde{W}_1 x_0\|_2 \\ &\leq L_\phi \sigma_{k_1+1}^{(1)} \|x_0\|_2. \end{aligned}$$

Inductive Step.

Assume the inductive bound holds for layer $i-1$. For layer i:

$$\begin{aligned} \|x_i - \tilde{x}_i\|_2 &= \|\phi(W_i x_{i-1}) - \phi(\tilde{W}_i \tilde{x}_{i-1})\|_2 \\ &\leq L_\phi \|W_i x_{i-1} - \tilde{W}_i \tilde{x}_{i-1}\|_2 \\ &\leq L_\phi \left(\|W_i (x_{i-1} - \tilde{x}_{i-1})\|_2 + \|(W_i - \tilde{W}_i)\tilde{x}_{i-1}\|_2 \right) \\ &\leq L_\phi \left(\|W_i\|_2 \|x_{i-1} - \tilde{x}_{i-1}\|_2 + \sigma_{k_i+1}^{(i)} \|\tilde{x}_{i-1}\|_2 \right). \end{aligned}$$

We can bound $\|\tilde{x}_{i-1}\|_2$ using the triangle inequality:

$$\|\tilde{x}_{i-1}\|_2 \leq \|x_{i-1}\|_2 + \|x_{i-1} - \tilde{x}_{i-1}\|_2.$$

Assuming that $\|x_{i-1}\|_2$ is bounded (which is reasonable in practice due to normalization techniques), and applying the inductive hypothesis, we can express $\|x_i - \tilde{x}_i\|_2$ in terms of the accumulated errors up to layer i.

By recursively applying this inequality and summing over all layers, we obtain the bound stated in Theorem 3.

□

F.1.3 Note on Activation Functions and the Lipschitz Constant

It is important to note that Theorem 3 assumes the activation function ϕ has a Lipschitz constant L_ϕ, which reflects how much the function can amplify differences in its input. For activation functions like ReLU, which are 1-Lipschitz, the error bound simplifies and indicates minimal error amplification through the activation layers.

However, the LLaMA model family uses activation functions such as SwiGLU and GELU, whose derivatives can exceed 1, making them not 1-Lipschitz. For networks employing such activation functions, the error propagation bound in Theorem 2 is adjusted by incorporating a Lipschitz constant L_ϕ, which may be greater than 1. This adjustment accounts for the potential additional error amplification introduced by the activation functions.

APPENDIX G

ADDITIONAL RESULTS OF LORC

G.1 Reconstruction Error of Matrix SVD

Given a pre-trained LLM, we conduct layer-wise weight matrix decomposition and reconstruction. We found that these matrices are low-rank and therefore can be reconstructed with low-dimension matrices, resulting in minimal reconstruction error. It means instead complex eviction policy design at the token level, we can turn to the attention level to develop a model and task agnostic KV cache compression method. We present the relative reconstruction error in Figure G.1, which is computed as below.

Given an original matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and its reconstructed approximation $\hat{\mathbf{A}}$ obtained via low-rank decomposition as $\hat{\mathbf{A}} = \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^\top$, the relative reconstruction error err is defined as:

$$\text{err} = \frac{\|\mathbf{A} - \hat{\mathbf{A}}\|_F}{\|\mathbf{A}\|_F},$$

where $\|\cdot\|_F$ denotes the Frobenius norm, defined as $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} X_{ij}^2}$.

Table G.1: Model Architectures.

Model	Attention	Layers	Heads	KV Heads	Head Dimension	Model Dimension	Weight Shape
LLaMA-2-13B	MHA	40	40	40	128	5120	5120×5120
LLaMA-3-Instruct-8B	GQA	32	32	8	128	4096	4096×1024
LLaMA-3-Instruct-70B	GQA	80	64	8	128	8192	8192×1024

G.2 Latency Analysis

The latency analysis is shown in Table Table G.2. Note that sensitivity calculation only takes place once for a given model, and the SVD processing is a one-time implementation during the model initialization stage. Such latency will not affect inference speed. Compared to the whole inference duration, the latency incurred by these processes is negligible. For example,

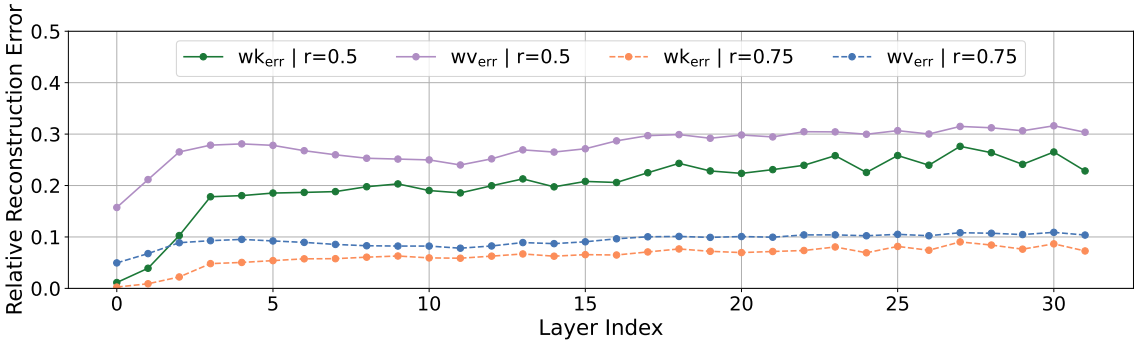


Figure G.1: Layerwise relative reconstruction errors. wk_{err} and wv_{err} denote the relative difference between the original key/value matrices and their corresponding low-rank approximations measured using the Frobinus norm. The compression ratio is computed as $r = \frac{d_c}{N_h \times d_h}$, where N_h is the number of attention heads and d_h, d_c is the original and compressed hidden dimensions respectively.

Table G.2: Latency analysis for sensitivity calculation and SVD processing.

Model	Computing Resource	Sensitivity Calculation (s)	SVD Processing (s)
LLaMA-3-8B-Instruct	NVIDIA H100 80GB HBM3 $\times$ 1	30.6	33.3
LLaMA-3-70B-Instruct	NVIDIA H100 80GB HBM3 $\times$ 8	12.2	14.3

the LLaMA-3-70B-Instruct model requires approximately 1 hour and 8 minutes to process 1,000 summaries from the XSUM dataset with a batch size of 32 and a sequence length of 8,000. The combined latency introduced by sensitivity calculations and SVD processing represents only 0.6% of the total inference time.

G.3 Throughput Analysis

Our method involves additional computations to recover the compressed cache and manage RoPE. To address this, we developed a customized kernel that fuses cache reconstruction and rotation operations. This approach minimizes memory transfers and computational overhead, enhancing the overall throughput.

Following H2O [150], we conducted throughput experiments with fixed input and output sequence lengths using the LLaMA-3-70B-Instruct model on a node equipped with eight NVIDIA H100 80GB HBM3 GPUs. The results below indicate that our engineering efforts

can streamline the attention computation with LoRC compression, thereby achieving higher throughput compared to full cache scenarios.

Table G.3: Throughput of LLaMA-3-70B-Instruct with and without LoRC compression.

Input Length	Output Length	Batch Size	Full Cache Throughput	LoRC-60% Throughput	Speedup
1024	2048	32	52.75 tokens/s	60.08 tokens/s	$\times$ 1.14
1024	4096	32	78.66 tokens/s	98.74 tokens/s	$\times$ 1.26

G.4 Comparison with Other KV Cache Compression Methods

We compare LoRC with a token-eviction method – H2O [150] and a quantization method – KIVI [170] in this section. We conduct experiments using LLaMA-3-8B-Instruct. For H2O and LoRC, we keep the same 60% KV cache budget. For KIVI, we use the KIVI-4-bit implementation. We evaluate accuracy on BoolQ and OpenBook QA, and Rouge-Lsum for XSum. The results below show LoRC can preserve a better performance compared to token-eviction and quantization methods when KV cache is aggressively compressed.

Table G.4: Performance comparison across different KV cache compression methods.

Method	BoolQ	XSum	OpenBookQA
Full Cache	81.6	11.6	78.0
H2O [150]	76.4	10.5	75.1
KIVI [170]	77.6	10.3	74.8
LoRC	79.2	11.2	75.7

For throughput comparison with the other methods, we use the metric seconds per iteration (s/it) on XSum. We adhered to the same 60% KV cache budget for both H2O and LoRC. In an effort to demonstrate compatibility with other KV cache compression methods, we integrated LoRC with both H2O and standard 8-bit quantization. Specifically, we allocated a 70% KV cache budget to LoRC and 85% to H2O, resulting in approximately a 60% overall cache budget. A similar configuration was used for the combination of LoRC with 8-bit quantization.

Table G.5: Throughput and performance comparison of different methods..

	Full Cache	H2O	KIVI	LoRC	LoRC w/ H2O	LoRC w/ 8-bit
Performance	11.6	10.5	10.3	11.2	11.0	10.9
s/it	228.5	192.7	186.6	203.4	195.9	193.7

Our results show that LoRC achieves superior performance compared to existing compression methods while maintaining competitive throughput. While the SVD-based computations introduce some overhead, LoRC still has a higher throughput than its full cache baseline because we deployed a customized kernel to streamline the attention computation. It is important to note that our primary objective was not to develop a Pareto-optimal method (simultaneously optimizing both performance and throughput), but rather to introduce an orthogonal approach to KV cache compression and establish the progressive compression strategy.

Additionally, the integration of LoRC with both token-eviction and quantization methods demonstrates its compatibility with existing compression methods. With only a slight performance trade-off (11.0/10.9 vs 11.2), these hybrid configurations achieve better throughput compared to using LoRC alone, which shows the potential of combining LoRC with orthogonal compression approaches.

G.5 Enhancements in memory-constrained deployment scenarios

Table G.6: LLaMA-3-8B-Instruct model on a single GPU with 24GB memory.

Max Sequence Length	Batch Size	8B w/ Full Cache	8B w/ LoRC-60%
2048	20	OOM	22.12 G

Table G.7: LLaMA-3-70B-Instruct model on 8 GPUs with 24GB memory.

Max Sequence Length	Batch Size	70B w/ Full Cache	70B w/ LoRC-60%
2048	8	OOM	22.97 G

LoRC demonstrates significant benefits in memory-constrained deployment scenarios. To further validate this, we provide additional experiments here. These results demonstrate that LoRC enables the deployment of models that would otherwise be impossible to run with full KV cache on consumer GPUs.

APPENDIX H

ADDITIONAL DISCUSSION OF LORC

H.1 Adjusted Position Embedding

[171] propose a rotary position embedding (RoPE) and it has been used in most recent LLMs. Applying RoPE to self-attention gives

$$q_m^T k_n = (R^d_{\Theta,m} W_q^T x_m)^T (R^d_{\Theta,n} W_k^T x_n) = x^T W_q R^d_{\Theta,n-m} W_k^T x_n, \tag{H.1}$$

where Θ is a pre-defined rotary matrix, m and n denotes the token position. In practice, the rotation matrix $R^d_{\Theta,n-m}$ is decomposed as $(R^d_{\Theta,m})^T$ and $R^d_{\Theta,n}$ to rotate the query and key separately, and the KV cache stores the rotated keys. To ensure that our compressed keys are compatible with the rotary operation, we adjust the position embedding pipeline. Specifically, we store the compressed keys $X(\Sigma V^\top)^\top_{D\times d_c}$ in cache, while incorporating the rotation and key projection into the query computation to streamline the process.

RoPE does bring additional computations to LoRC. To address this, we follow H2O [150] and FastGen [151] to develop a customized kernel that fuses cache reconstruction and rotation operations. This approach minimizes memory transfers and computational overhead, enhancing the overall throughput.

Specifically, we leverage Triton [172] to implement a fused kernel that optimizes memory access and computation. With this kernel, we are able to combine key reconstruction from compressed representation and apply RoPE rotations on-the-fly. It also streamlines memory access because the single data load from global memory for compressed keys requires minimal data movement.

In a complete RoPE workflow, there are two stages:

- **Caching:** We project keys to low-rank space without position embedding and store

only the compressed representation.

- **Attention computation:** Within the customized kernel, we load compressed keys into shared memory, reconstruct and apply RoPE in a fused operation, and then compute attention scores with position-aware keys.

With these engineering efforts, we achieve higher throughput compared to baseline configurations.

H.2 Contrasting Sensitivity Trends Between Training-stage and Inference-stage Compression

It is observed that the sensitivity trends are contrasting between training-stage compression [173, 174, 175] and inference-stage compression (LoRC). This is because forward propagation amplifies early-layer compression errors during inference, while backward propagation amplifies later-layer compression effects during training.

To provide a detailed discussion, we can compare the error propagation mechanisms. In our method, errors from compressed earlier layers propagate forward through the network. As shown in our Theorem Theorem 4, the error at network output is bounded by Eq. Equation 5.16, where earlier layer errors get amplified more because they pass through more subsequent transformations and nonlinearities.

For those training-stage compression methods as introduced in [4-6], their rationale is that gradient errors propagate backward through the network. For example, DropBP [6] measures the layer sensitivity by:

$$S_l = \sum_i \left(\|\nabla \mathbf{W}_i\|_2 - \|\nabla \mathbf{W}_i^{(l)}\|_2 \right)^2$$

where S_l denotes the sensitivity of the l-th layer. Here, $\nabla \mathbf{W}_i$ represents the parameter gradient of the i-th layer when no layers are dropped, while $\nabla \mathbf{W}_i^{(l)}$ denotes the parameter gradient of the i-th layer when the l-th layer is dropped during backward propagation. It

shows that later layers have higher impact on gradients propagating backward to all previous layers.

APPENDIX I

IMPLEMENTATION DETAILS OF DORM

I.1 Dataset Statistics and Preprocessing

We evaluate our method on multiple preference datasets, each with distinct characteristics. Table Table I.1 summarizes the dataset sizes and key attributes.

Table I.1: Summary of the preference datasets used in our experiments.

Dataset	Number of Attributes	**Train Set Size**	**Validation Set Size**	**Annotation Type**	**Primary Use**
HelpSteer2 [84]	5	20k	{0.7k, 1k, 2k}	Human-annotated	General alignment
OpenAssistant2 [90]	4	{5k, 20k}	{0.7k, 1k, 2k}	Human-annotated	Assistant-style conversations
Magpie-QWen-2.5 [91]	1	{10k, 32k}	{0.7k, 1k, 2k}	Synthetic	Fine-tuning
OffsetBias [92]	1	{10k, 16k}	{0.7k, 1k, 2k}	Human-annotated	Bias mitigation
WildGuard [93]	1	{5k, 12k}	{0.7k, 1k, 2k}	Adversarial	Safety evaluation

Normalization of Rating Scales. These datasets use varying rating scales, which can introduce inconsistencies in preference modeling. For instance, HelpSteer2 employs a scale of 0-4, whereas other datasets use binary or continuous scores in different ranges. To standardize these ratings, we apply a linear transformation to convert all scores to a common 0-4 scale. Binary labels are mapped such that the preferred response corresponds to 4 and the non-preferred to 0.

Merging Reward Attributes. Some datasets contain overlapping but non-identical preference objectives. Since these datasets follow different annotation rubrics and evaluation protocols, we adopt the approach of [98] and treat such objectives separately to maintain consistency. This ensures that preference signals remain distinct, preventing biases introduced by differing annotation standards. As a result, our reward model incorporates a 12-attribute head.

These preprocessing steps ensure that our dataset is well-structured for reward modeling and policy optimization, reducing the impact of inconsistencies across different sources.

Table I.2: Models and their corresponding training dataset sizes.

Model	Number of Training Data	RewardBench Overall Score
Llama3-70B-SteerLM-RM [84]	20k	88.8
URM-Llama3-8B [75]	20k	89.9
ArmoRM-Llama3-8B-v0.1 [98] †	585.4k	90.4
Pair-preference-model-LLaMA3-8B [63] †	585.4k	87.1
Internlm2-20B-reward [99] †	2.4 million	90.2
DORM	50k	90.5

I.2 Training Recipe Comparison

Table Table I.2 presents the training data sizes of baseline models in our main experiments. achieves strong performance on RewardBench while using only 50k training data (with 5k validation set), which is $10\times$ less than *ArmoRM-Llama3-8B-v0.1* and *Pair-preference-model-LLama3-8B*, and $40\times$ less than *Internlm2-20B-reward*. Despite the significantly smaller dataset size, our model attains a RewardBench overall score of 90.5, outperforming models trained on substantially larger datasets.

This data efficiency stems from our quality-aware preference weighting strategy and dynamic weighting in reward modeling. By prioritizing informative and reliable signals, our method reduces reliance on large-scale preference data. Additionally, bilevel optimization ensures effective use of high-value samples, mitigating performance degradation from noise or redundancy. These techniques allow our model to achieve strong performance with significantly fewer training samples.

I.3 Implementation Details

Reward Model Training. Our reward model is initialized with Mistral NeMo 12B-Instruct [102] and fine-tuned using a 12-attribute head. Table Table I.3 outlines the hyperparameters used.

Bilevel Optimization. We initialize the proxy u_0 from θ_0, and initialize w_0 being uniform. The learning rate is$\eta_\theta = \eta_u = 3e^-6, \eta_w = 0.001$ The coefficients are set to $\alpha = 10.0, \beta =$

Table I.3: Training hyperparameters for reward model fine-tuning.

Hyperparameter	**Value**	**Notes**
Reward Head Dimension	12	Correspond to Number of Reward Attributes
Batch Size	4	Per GPU
Learning Rate	3×10^{-6}	Cosine decay
Weight Decay	0.01	Applied to Adam optimizer
Betas	0.9, 0.98	Applied to Adam optimizer
Warm-up Steps	200	Cosine scheduler
Mixed Precision	BF16	Reduces memory consumption
Number of Epochs	3	Maximum Training Epochs

1.0. We use Adam [176] as the optimizer for both levels.

The training is conducted on two nodes (16 A100-80G GPUs in total), and it takes about 10.4 hours for 3-epoch training. During inference, we implement a Bayesian hyperparameter search to determine the projection vector λ on the validation set. This vector is used to compute a weighted sum from the multi-attribute outputs of our reward model, enabling evaluation against the binary labels of RewardBench.

APPENDIX J

ADDITIONAL RESULTS OF DORM

J.1 Results with Different Data Size

We investigate the impact of the training/validation set size on model performance. Table Table J.1 summarizes the results.

Table J.1: Effect of validation set size on model performance.

Training Size	Validation Size	Chat	Chat Hard	Reasoning	Safety	Overall
50k	3.5k	94.7	84.8	88.5	92.9	90.2
50k	5k	95.6	83.4	89.8	93.1	90.5
50k	10k	96.8	84.9	89.1	95.3	91.5
100k	3.5k	96.9	85.2	88.7	95.1	91.5
100k	5k	96.8	85.5	89.2	96.0	91.9
100k	10k	97.1	85.6	90.6	94.9	92.1

Larger validation sets improve model performance. As the validation set size increases from 3.5k to 10k, the overall performance improves consistently across all training configurations. For instance, with 50k training samples, expanding the validation set from 3.5k to 10k leads to an increase in the overall score from 90.2 to 91.5, with notable gains in safety (92.9 to 95.3). This suggests that a larger validation set provides a more reliable signal for optimizing data weights, contributing to better model performance.

Increased training data enhances robustness. Expanding the training set from 50k to 100k also yields consistent improvements in overall performance, particularly in safety and reasoning. With a 10k validation set, the overall score increases from 91.5 (50k training) to 92.1 (100k training), while safety improves from 95.3 to 94.9. These results indicate that additional training data strengthens the model's ability to generalize across different tasks.

J.2 Study on Uncertainty Measurement

In subsection 3.4.1, we use MC dropout for epistemic uncertainty estimation. Our method is compatible with other uncertainty estimation techniques. We include sigmoid-based and ensemble-based methods for comparison and illustrate our rational for using MC dropout.

Table J.2: Ablation study results on uncertainty-based weighting.

Method	**Chat**	**Chat Hard**	**Reasoning**	**Safety**	**Overall**
Baseline	90.7	72.8	86.7	91.1	85.3
MC Dropout	91.9	77.0	85.2	91.6	86.4
Ensemble	87.7	74.6	89.8	92.5	86.1
Sigmoid	86.8	72.8	89.2	91.9	85.2

MC dropout achieves the highest overall score of 86.4, outperforming the baseline 85.3 and other uncertainty-estimation methods. It provides balanced improvements across all tasks, particularly in chat hard $+4.2$ and chat $+1.2$. While ensemble-based estimation achieves slightly better reasoning and safety scores, its overall performance remains lower.

MC dropout is also computationally efficient, requiring only stochastic forward passes within a single model, whereas ensemble methods require multiple model instances. Although the sigmoid-based approach is computationally lightweight, it does not provide sufficient performance gains.

In summary, our method supports various uncertainty estimation techniques. MC dropout is preferred due to its strong empirical performance and efficiency.

J.2.1 Why Epistemic Uncertainty and Disagreements Rather Than Aleatory Uncertainty?

In our quality estimation for the preference data, we consider uncertainty to identify which data points are more informative for the reward model. There are two common forms of uncertainty: epistemic and aleatory [86]. Epistemic uncertainty arises from the model's lack of knowledge about the underlying preference function, while aleatory uncertainty is related to inherent randomness or noise in the data generation process. Our quality estimation

focuses on epistemic uncertainty and disagreements rather than aleatory uncertainty for the following reasons:

1. **Epistemic uncertainty is reducible through data or model improvements:** Epistemic uncertainty reflects gaps in the model's current representation. If a data point causes the model to be uncertain, it suggests the model does not fully understand the underlying preference structure at that point. Prioritizing such data helps refine model parameters. In contrast, aleatory uncertainty stems from inherent randomness and cannot be mitigated through training.

2. **Disagreements highlight label reliability issues:** Discrepancies between model predictions and labels indicate potential label noise or annotation errors. These disagreements help us identify data that may be mislabeled or difficult for the model. Adjusting data weights based on these disagreements allows the model to focus on data that can provide more reliable learning signals.

3. **Aleatory uncertainty does not provide actionable insights for weighting:** Since aleatory uncertainty persists regardless of training, incorporating it into data weighting does not improve learning. Epistemic uncertainty and disagreements, however, offer actionable insights for data selection and model refinement.

For these reasons, our quality estimation leverages epistemic uncertainty and disagreements to guide the model toward more informative data, improving training efficiency and alignment.

REFERENCES

[1] OpenAI, *Gpt-4 technical report*, 2023.

[2] Anthropic, *Claude sonnet*, https://www.anthropic.com/claude/sonnet, Accessed: 2025-03-23, 2024.

[3] G. Team *et al.*, "Gemini: A family of highly capable multimodal models", *arXiv preprint arXiv:2312.11805*, 2023.

[4] DeepSeek-AI, *Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning*, 2025. arXiv: 2501.12948 [cs.CL].

[5] Y. Gao, D. Alon, and D. Metzler, "Impact of preference noise on the alignment performance of generative language models", *arXiv preprint arXiv:2404.09824*, 2024.

[6] A. Awasthi, S. Ghosh, R. Goyal, and S. Sarawagi, "Learning from rules generalizing labeled exemplars", in *International Conference on Learning Representations*, 2020.

[7] D. Mekala and J. Shang, "Contextualized weak supervision for text classification", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 323–333.

[8] Y. Meng *et al.*, "Text classification using label names only: A language model self-training approach", in *EMNLP*, Nov. 2020, pp. 9006–9017.

[9] Y. Yu, S. Zuo, H. Jiang, W. Ren, T. Zhao, and C. Zhang, "Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach", in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 1063–1077.

[10] W. Zhou *et al.*, "Nero: A neural rule grounding framework for label-efficient relation extraction", in *The Web Conference*, 2020, pp. 2166–2176.

[11] P. Lison, J. Barnes, A. Hubin, and S. Touileb, "Named entity recognition without labelled data: A weak supervision approach", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Jul. 2020, pp. 1518–1533.

[12] E. Safranchik, S. Luo, and S. Bach, "Weakly supervised sequence tagging from noisy rules", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5570–5578.

[13] Y. Li, P. Shetty, L. Liu, C. Zhang, and L. Song, "BERTifying the hidden Markov model for multi-source weakly supervised named entity recognition", in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 6178–6190.

[14] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision", in *VLDB*, vol. 11, 2017, p. 269.

[15] B. Hancock, P. Varma, S. Wang, M. Bringmann, P. Liang, and C. Ré, "Training classifiers with natural language explanations", in *ACL*, Jul. 2018, pp. 1884–1895.

[16] P. Varma and C. Ré, "Snuba: Automating weak supervision to label training data", in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, NIH Public Access, vol. 12, 2018, p. 223.

[17] J. Zhang, C.-Y. Hsieh, Y. Yu, C. Zhang, and A. Ratner, "A survey on programmatic weak supervision", *arXiv preprint arXiv:2202.05433*, 2022.

[18] B. Boecking, W. Neiswanger, E. Xing, and A. Dubrawski, "Interactive weak supervision: Learning useful heuristics for data labeling", in *ICLR*, 2021.

[19] S. Galhotra, B. Golshan, and W.-C. Tan, "Adaptive rule discovery for labeling text data", in *SIGMOD*, 2021, pp. 2217–2225.

[20] J. Zhang *et al.*, "WRENCH: A comprehensive benchmark for weak supervision", in *NeurIPS*, 2021.

[21] J. Li, H. Ding, J. Shang, J. McAuley, and Z. Feng, "Weakly supervised named entity tagging with learnable logical rules", in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Association for Computational Linguistics, Aug. 2021, pp. 4568–4581.

[22] X. Zhao, H. Ding, and Z. Feng, "GLaRA: Graph-based labeling rule augmentation for weakly supervised named entity recognition", in *EACL*, Association for Computational Linguistics, Apr. 2021, pp. 3636–3649.

[23] A. Holub, P. Perona, and M. C. Burl, "Entropy-based active learning for object recognition", in *CVPR Workshops*, IEEE, 2008, pp. 1–8.

[24] Y. Shen, H. Yun, Z. Lipton, Y. Kronrod, and A. Anandkumar, "Deep active learning for named entity recognition", in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 252–256.

[25] R. Zhang, Y. Yu, and C. Zhang, "Seqmix: Augmenting active sequence labeling via sequence mixup", in *EMNLP*, 2020, pp. 8566–8579.

[26] L. Ein-Dor *et al.*, "Active Learning for BERT: An Empirical Study", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 7949–7962.

[27] K. Margatina, G. Vernikos, L. Barrault, and N. Aletras, "Active learning by acquiring contrastive examples", in *EMNLP*, Nov. 2021, pp. 650–663.

[28] Y. Yu, L. Kong, J. Zhang, R. Zhang, and C. Zhang, "Actune: Uncertainty-based active self-training for active fine-tuning of pretrained language models", *NAACL*, 2022.

[29] D. Choi, S. Evensen, Ç. Demiralp, and E. Hruschka, "Tagruler: Interactive tool for span-level data programming by demonstration", in *WWW*, 2021, pp. 673–677.

[30] C.-Y. Hsieh, J. Zhang, and A. Ratner, "Nemo: Guiding and contextualizing weak supervision for interactive data programming", *arXiv*, 2022.

[31] T. Brown *et al.*, "Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[32] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing", *arXiv*, 2021.

[33] S. Hu, N. Ding, H. Wang, Z. Liu, J. Li, and M. Sun, "Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification", *arXiv preprint arXiv:2108.02035*, 2021.

[34] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification", *arXiv preprint arXiv:2105.11259*, 2021.

[35] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference", in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online: Association for Computational Linguistics, Apr. 2021, pp. 255–269.

[36] T. Schick and H. Schütze, "It's not just size that matters: Small language models are also few-shot learners", in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2021, pp. 2339–2352.

[37] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning", *arXiv preprint arXiv:2104.08691*, 2021.

[38] X. Chen *et al.*, "Adaprompt: Adaptive prompt-based finetuning for relation extraction", *arXiv preprint arXiv:2104.07650*, 2021.

[39] Z.-Y. Dou, P. Liu, H. Hayashi, Z. Jiang, and G. Neubig, "GSum: A general framework for guided neural abstractive summarization", in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2021, pp. 4830–4842.

[40] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation", in *ACL*, Aug. 2021, pp. 4582–4597.

[41] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners", in *ACL*, Aug. 2021, pp. 3816–3830.

[42] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks", *arXiv*, 2021.

[43] X. Liu *et al.*, "Gpt understands, too", *arXiv preprint arXiv:2103.10385*, 2021.

[44] C. Liang *et al.*, "Bond: Bert-assisted open-domain named entity recognition with distant supervision", in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 1054–1064, ISBN: 9781450379984.

[45] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing, "Harnessing deep neural networks with logic rules", in *ACL*, Aug. 2016, pp. 2410–2420.

[46] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost", *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[47] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks", in *ICML Workshop*, 2013, p. 896.

[48] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis", in *International conference on machine learning*, PMLR, 2016, pp. 478–487.

[49] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling", in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 35–45.

[50] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification", *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.

[51] M. Krallinger, O. Rabal, S. A. Akhondi, *et al.*, "Overview of the biocreative VI chemical-protein interaction track", in *BioCreative evaluation Workshop*, vol. 1, 2017, pp. 141–146.

[52] E. Perez, D. Kiela, and K. Cho, "True few-shot learning with language models", *arXiv preprint arXiv:2105.11447*, 2021.

[53] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach", *arXiv*, 2019.

[54] J. Lee *et al.*, "Biobert: A pre-trained biomedical language representation model for biomedical text mining", *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[55] J. L. Fleiss, "Measuring nominal scale agreement among many raters.", *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.", *Journal of machine learning research*, vol. 9, no. 11, 2008.

[57] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback", *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[58] Y. Zhao, R. Joshi, T. Liu, M. Khalman, M. Saleh, and P. J. Liu, "Slic-hf: Sequence likelihood calibration with human feedback", *arXiv preprint arXiv:2305.10425*, 2023.

[59] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model", in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[60] L. Zheng *et al.*, *Judging llm-as-a-judge with mt-bench and chatbot arena*, 2023. arXiv: 2306.05685 [cs.CL].

[61] B. Wang *et al.*, "Secrets of rlhf in large language models part ii: Reward modeling", *arXiv preprint arXiv:2401.06080*, 2024.

[62] C. Y. Liu and L. Zeng, *Skywork reward model series*, https://huggingface.co/Skywork, Sep. 2024.

[63] H. Dong *et al.*, "Rlhf workflow: From reward modeling to online rlhf", *arXiv preprint arXiv:2405.07863*, 2024.

[64] R. Yu, Y. Wang, X. Jiao, Y. Zhang, and J. T. Kwok, "Direct alignment of language models via quality-aware self-refinement", *arXiv preprint arXiv:2405.21040*, 2024.

[65] J. Cheng, G. Xiong, X. Dai, Q. Miao, Y. Lv, and F.-Y. Wang, "Rime: Robust preference-based reinforcement learning with noisy preferences", *arXiv preprint arXiv:2402.17257*, 2024.

[66] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", in *international conference on machine learning*, PMLR, 2016, pp. 1050–1059.

[67] A. Dubey *et al.*, "The llama 3 herd of models", *arXiv preprint arXiv:2407.21783*, 2024.

[68] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, and T. Lillicrap, "Alayrac, et al. gemini 1.5: Unlocking multimodal understanding across millions of tokens of context", *arXiv preprint arXiv:2403.05530*, vol. 1, no. 3, p. 5, 2024.

[69] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming", *Advances in Neural Information Processing Systems*, vol. 35, pp. 9460–9471, 2022.

[70] T. Liu *et al.*, "Rrm: Robust reward model training mitigates reward hacking", *arXiv preprint arXiv:2409.13156*, 2024.

[71] L. Engstrom *et al.*, "Implementation matters in deep policy gradients: A case study on ppo and trpo", *arXiv preprint arXiv:2005.12729*, 2020.

[72] X. Wang *et al.*, "Deep reinforcement learning: A survey", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2022.

[73] H. Lee *et al.*, "Rlaif: Scaling reinforcement learning from human feedback with ai feedback", *arXiv preprint arXiv:2309.00267*, 2023.

[74] Y. Bai *et al.*, *Training a helpful and harmless assistant with reinforcement learning from human feedback*, 2022. arXiv: 2204.05862 [cs.CL].

[75] X. Lou, D. Yan, W. Shen, Y. Yan, J. Xie, and J. Zhang, "Uncertainty-aware reward model: Teaching reward models to know what is unknown", *arXiv preprint arXiv:2410.00847*, 2024.

[76] J. Zhang, K. Bao, Y. Zhang, W. Wang, F. Feng, and X. He, "Is chatgpt fair for recommendation", *Evaluating Fairness in Large Language Model Recommendation*, 2023.

[77] G. H. Chen, S. Chen, Z. Liu, F. Jiang, and B. Wang, "Humans or llms as the judge? a study on judgement biases", *arXiv preprint arXiv:2402.10669*, 2024.

[78] Y. Wang, X. Wu, H.-T. Wu, Z. Tao, and Y. Fang, "Do large language models rank fairly? an empirical study on the fairness of llms as rankers", *arXiv preprint arXiv:2404.03192*, 2024.

[79] Y. Bai *et al.*, "Constitutional ai: Harmlessness from ai feedback", *arXiv preprint arXiv:2212.08073*, 2022.

[80] B. Zhu, M. Jordan, and J. Jiao, "Principled reinforcement learning with human feedback from pairwise or k-wise comparisons", in *International Conference on Machine Learning*, PMLR, 2023, pp. 43 037–43 067.

[81] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons", *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[82] H. Wang *et al.*, "Arithmetic control of llms for diverse user preferences: Directional preference alignment with multi-objective rewards", *arXiv preprint arXiv:2402.18571*, 2024.

[83] Z. Wang *et al.*, "HelpSteer: Multi-attribute helpfulness dataset for SteerLM", in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 3371–3384.

[84] Z. Wang *et al.*, *Helpsteer2: Open-source dataset for training top-performing reward models*, 2024. arXiv: 2406.08673.

[85] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[86] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods", *Machine learning*, vol. 110, no. 3, pp. 457–506, 2021.

[87] J. Kwon, D. Kwon, S. Wright, and R. D. Nowak, "A fully first-order method for stochastic bilevel optimization", in *International Conference on Machine Learning*, PMLR, 2023, pp. 18 083–18 113.

[88] Z. Lu and S. Mei, "First-order penalty methods for bilevel optimization", *SIAM Journal on Optimization*, vol. 34, no. 2, pp. 1937–1969, 2024.

[89] Z. Wang *et al.*, *Helpsteer: Multi-attribute helpfulness dataset for steerlm*, 2023. arXiv: 2311.09528 [cs.CL].

[90] A. Köpf *et al.*, "Openassistant conversations-democratizing large language model alignment", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[91] Z. Xu *et al.*, "Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing", *arXiv preprint arXiv:2406.08464*, 2024.

[92] J. Park, S. Jwa, M. Ren, D. Kim, and S. Choi, "Offsetbias: Leveraging debiased data for tuning evaluators", *arXiv preprint arXiv:2407.06551*, 2024.

[93] S. Han *et al.*, *Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms*, 2024. arXiv: 2406.18495 [cs.CL].

[94] J. Achiam *et al.*, "Gpt-4 technical report", *arXiv preprint arXiv:2303.08774*, 2023.

[95] A. Hurst *et al.*, "Gpt-4o system card", *arXiv preprint arXiv:2410.21276*, 2024.

[96] T. Wang *et al.*, "Self-taught evaluators", *arXiv preprint arXiv:2408.02666*, 2024.

[97] N. Stiennon *et al.*, "Learning to summarize with human feedback", *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[98] H. Wang, W. Xiong, T. Xie, H. Zhao, and T. Zhang, "Interpretable preferences via multi-objective reward modeling and mixture-of-experts", *arXiv preprint arXiv:2406.12845*, 2024.

[99] Z. Cai *et al.*, *Internlm2 technical report*, 2024. arXiv: 2403.17297 [cs.CL].

[100] N. Lambert *et al.*, *Rewardbench: Evaluating reward models for language modeling*, 2024. arXiv: 2403.13787 [cs.LG].

[101] G. Cui *et al.*, *Ultrafeedback: Boosting language models with high-quality feedback*, 2023. arXiv: 2310.01377 [cs.CL].

[102] NVIDIA and Mistral AI, *Mistral-NeMo-12B-Instruct*, https://huggingface.co/nvidia/Mistral-NeMo-12B-Instruct.

[103] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression", in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541.

[104] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network", *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[105] A. K. Menon, A. S. Rawat, S. Reddi, S. Kim, and S. Kumar, "A statistical perspective on distillation", in *International Conference on Machine Learning*, PMLR, 2021, pp. 7632–7642.

[106] S. Stanton, P. Izmailov, P. Kirichenko, A. A. Alemi, and A. G. Wilson, "Does knowledge distillation really work?", *Advances in Neural Information Processing Systems*, vol. 34, pp. 6906–6919, 2021.

[107] J. C. White and R. Cotterell, "Examining the inductive bias of neural language models with artificial languages", in *Annual Meeting of the Association for Computational Linguistics*, 2021.

[108] B. Liu, S. Zhang, G. Song, H. You, and Y. Liu, "Rectifying the data bias in knowledge distillation", in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1477–1486.

[109] M. Lukasik, S. Bhojanapalli, A. K. Menon, and S. Kumar, "Teacher's pet: Understanding and mitigating biases in distillation", *arXiv preprint arXiv:2106.10494*, 2021.

[110] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[111] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[112] Z. Leng *et al.*, "Polyloss: A polynomial expansion perspective of classification loss functions", in *International Conference on Learning Representations*, 2022.

[113] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation", *arXiv preprint arXiv:1910.10699*, 2019.

[114] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3903–3911.

[115] A. Jafari, M. Rezagholizadeh, P. Sharma, and A. Ghodsi, "Annealing knowledge distillation", in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 2493–2504.

[116] P. Chen, S. Liu, H. Zhao, and J. Jia, "Distilling knowledge via knowledge review", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5008–5017.

[117] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation", in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2022, pp. 11 953–11 962.

[118] Y. Ren, S. Guo, and D. J. Sutherland, "Better supervisory signals by observing learning paths", *arXiv preprint arXiv:2203.02485*, 2022.

[119] W. Zhou, C. Xu, and J. McAuley, "Bert learns to teach: Knowledge distillation with meta learning", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 7037–7049.

[120] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation", *arXiv preprint arXiv:1606.07947*, 2016.

[121] Y. Wen, Z. Li, W. Du, and L. Mou, "F-divergence minimization for sequence-level knowledge distillation", *arXiv preprint arXiv:2307.15190*, 2023.

[122] Y. Gu, L. Dong, F. Wei, and M. Huang, "Knowledge distillation of large language models", *arXiv preprint arXiv:2306.08543*, 2023.

[123] R. Agarwal, N. Vieillard, P. Stanczyk, S. Ramos, M. Geist, and O. Bachem, "Gkd: Generalized knowledge distillation for auto-regressive sequence models", *arXiv preprint arXiv:2306.13649*, 2023.

[124] R. Zhang *et al.*, "Plad: Preference-based large language model distillation with pseudo-preference pairs", *arXiv preprint arXiv:2406.02886*, 2024.

[125] G. Ji and Z. Zhu, "Knowledge distillation in wide neural networks: Risk bound, data efficiency and imperfect teacher", *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 823–20 833, 2020.

[126] H. Zhou *et al.*, "Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective", *arXiv preprint arXiv:2102.00650*, 2021.

[127] D. Hsu, Z. Ji, M. Telgarsky, and L. Wang, "Generalization bounds via distillation", *arXiv preprint arXiv:2104.05641*, 2021.

[128] Z. Allen-Zhu and Y. Li, "Towards understanding ensemble, knowledge distillation and self-distillation in deep learning", in *ICLR*, 2023.

[129] S. Gonzalez and R. Miikkulainen, "Optimizing loss functions through multi-variate taylor polynomial parameterization", in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 305–313.

[130] OpenAI, *Chatgpt*, 2022.

[131] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?", in *NeurIPS*, 2019.

[132] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments", *arXiv preprint arXiv:1805.12471*, 2018.

[133] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference", *arXiv preprint arXiv:1704.05426*, 2017.

[134] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases", in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[135] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding", *arXiv preprint arXiv:1804.07461*, 2018.

[136] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova, "Boolq: Exploring the surprising difficulty of natural yes/no questions", *arXiv preprint arXiv:1905.10044*, 2019.

[137] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer.", *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[138] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

[139] S. Kullback, *Statistics and information theory*, 1959.

[140] T. Ishida, I. Yamane, T. Sakai, G. Niu, and M. Sugiyama, "Do we need zero training loss after achieving zero training error?", *arXiv preprint arXiv:2002.08709*, 2020.

[141] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways", *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[142] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models", *arXiv preprint arXiv:2307.09288*, 2023.

[143] A. Vaswani *et al.*, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[144] N. Shazeer, "Fast transformer decoding: One write-head is all you need", *arXiv preprint arXiv:1911.02150*, 2019.

[145] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints", *arXiv preprint arXiv:2305.13245*, 2023.

[146] L. Ribar, I. Chelombiev, L. Hudlass-Galley, C. Blake, C. Luschi, and D. Orr, "Sparq attention: Bandwidth-efficient llm inference", *arXiv preprint arXiv:2312.04985*, 2023.

[147] Z. Liu *et al.*, "Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[148] A. Liu *et al.*, "Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model", *arXiv preprint arXiv:2405.04434*, 2024.

[149] Y. Sheng *et al.*, "Flexgen: High-throughput generative inference of large language models with a single gpu", in *International Conference on Machine Learning*, PMLR, 2023, pp. 31 094–31 116.

[150] Z. Zhang *et al.*, "H2o: Heavy-hitter oracle for efficient generative inference of large language models", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[151] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao, "Model tells you what to discard: Adaptive kv cache compression for llms", *arXiv preprint arXiv:2310.01801*, 2023.

[152] R. Pope *et al.*, "Efficiently scaling transformer inference", *Proceedings of Machine Learning and Systems*, vol. 5, 2023.

[153] M. de Jong *et al.*, "Fido: Fusion-in-decoder optimized for stronger performance and faster inference", *arXiv preprint arXiv:2212.08153*, 2022.

[154] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering", *arXiv preprint arXiv:2007.01282*, 2020.

[155] C. Holmes *et al.*, "Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference", *arXiv preprint arXiv:2401.08671*, 2024.

[156] Y. Guan, Z. Li, J. Leng, Z. Lin, and M. Guo, "Transkimmer: Transformer learns to layer-wise skim", *arXiv preprint arXiv:2205.07324*, 2022.

[157] T. Sun *et al.*, "A simple hash-based early exiting approach for language understanding and generation", *arXiv preprint arXiv:2203.01670*, 2022.

[158] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit", *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 330–18 341, 2020.

[159] J. Mu, X. Li, and N. Goodman, "Learning to compress prompts with gist tokens", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[160] Y. Zhang *et al.*, "Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling", *arXiv preprint arXiv:2406.02069*, 2024.

[161] H. Yu, Z. Yang, S. Li, Y. Li, and J. Wu, "Effectively compress kv heads for llm", *arXiv preprint arXiv:2406.07056*, 2024.

[162] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization", *ArXiv*, vol. abs/1808.08745, 2018.

[163] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering", *arXiv preprint arXiv:1809.02789*, 2018.

[164] K. Cobbe *et al.*, "Training verifiers to solve math word problems", *arXiv preprint arXiv:2110.14168*, 2021.

[165] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries", in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[166] P. Sharma, J. T. Ash, and D. Misra, "The truth is in there: Improving reasoning in language models with layer-selective rank reduction", *ArXiv*, vol. abs/2312.13558, 2023.

[167] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization", in *International Conference on Learning Representations*, 2019.

[168] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library", *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[169] S. Boucheron, O. Bousquet, and G. Lugosi, "Theory of classification: A survey of some recent advances", *ESAIM: probability and statistics*, vol. 9, pp. 323–375, 2005.

[170] Z. Liu *et al.*, "Kivi: A tuning-free asymmetric 2bit quantization for kv cache", *arXiv preprint arXiv:2402.02750*, 2024.

[171] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding", *Neurocomputing*, vol. 568, p. 127 063, 2024.

[172] P. Tillet, H.-T. Kung, and D. Cox, "Triton: An intermediate language and compiler for tiled neural network computations", in *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, 2019, pp. 10–19.

[173] X. Liu *et al.*, "Gact: Activation compressed training for generic network architectures", in *International Conference on Machine Learning*, PMLR, 2022, pp. 14 139–14 152.

[174] S. Woo, S. Lee, and D. Jeon, "Alam: Averaged low-precision activation for memory-efficient training of transformer models", in *The Twelfth International Conference on Learning Representations*, 2024.

[175] S. Woo *et al.*, "Dropbp: Accelerating fine-tuning of large language models by dropping backward propagation", *arXiv preprint arXiv:2402.17812*, 2024.

[176] D. P. Kingma, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

VITA

Rongzhi Zhang received his B.E. degree from Zhejiang University in 2019 and his M.S degree from Georgia Institute of Technology in 2021. He then joined the Machine Learning Ph.D. program in the School of Computational Science and Engineering at Georgia Tech, advised by Dr. Chao Zhang.