UNCERTAINTY-AWARE AND DATA-EFFICIENT FINE-TUNING AND APPLICATION OF FOUNDATION MODELS

A Dissertation
Presented to
The Academic Faculty

By

Yinghao Li

In Partial Fulfillment
of the Requirements for the Degree
in the
School of Electrical and Computer Engineering
College of Engineering

Georgia Institute of Technology

April 2025

UNCERTAINTY-AWARE AND DATA-EFFICIENT FINE-TUNING AND APPLICATION OF FOUNDATION MODELS

Thesis committee:

Dr. Chao Zhang Computational Science and Engineering *Georgia Institute of Technology*	Dr. Srijan Kumar Computational Science and Engineering *Georgia Institute of Technology*
Dr. Rampi Ramprasad Materials Science and Engineering *Georgia Institute of Technology*	Dr. Victor Fung Computational Science and Engineering *Georgia Institute of Technology*
Dr. Tuo Zhao Industrial and Systems Engineering *Georgia Institute of Technology*	Dr. MohamadAli Torkamani Generative AI *Amazon Web Service*

Date approved: April 17, 2025

ACKNOWLEDGMENTS

I would like to express my profound gratitude to all those who have supported, guided, and encouraged me throughout my PhD journey.

First and foremost, I am deeply indebted to my advisor, Dr. Chao Zhang, my co-advisor, Dr. Rampi Ramprasad, and Prof. Le Song. Their mentorship was pivotal in shaping my research interests and approach, and I am especially grateful for Dr. Zhang's thoughtful guidance, which helped me navigate the intricacies of scientific inquiry. Their collective expertise, unwavering support, and commitment to intellectual rigor have been invaluable to my development as a researcher.

I also wish to thank my esteemed dissertation committee members: Dr. Chao Zhang, Dr. Rampi Ramprasad, Dr. Tuo Zhao, Dr. Srijan Kumar, Dr. Victor Fung, and Dr. Ali Torkamani. Your time, insight, and constructive feedback significantly influenced the direction and quality of my dissertation, and your dedication throughout this critical phase of my PhD is deeply appreciated.

I am further grateful for the mentorship of Dr. Prashant Shiralkar, Dr. Colin Lockard, Dr. Vianne Gao, Prof. Ying Zhang, and Prof. Matthieu R. Bloch. Your guidance and support have played a key role in shaping my research and professional growth.

My heartfelt thanks go out to my parents and to my friends, both those I met before my PhD and those who joined me along the way: Bingchen Du, Haotian Sun, Xiayan Ji, Yiliang Guo, Shengwei Lyu, Wendi Ren, Ciyuan Liu, Zhe Yu, Yang Yue, Entong Zhao, Shangqing Xu, Brian and JoAnn Boham, Chen Ling, Zekuan Liu, Xuan Zhang, Le Yu, Zhichao Hou, Ziheng Liu, Lama Moukheiber, Tracey West, and so many others. Your unwavering support, constant encouragement, and genuine friendship have been an extraordinary source of strength and perspective.

I am equally appreciative of my collaborators and lab mates: Haorui Wang, Rushi Qiang, Yue Yu, Zongyang Xia, Jerry Junyang Cheung, Yuchen Zhuang, Yuzhao Heng, Jieyu Zhang, Rongzhi Zhang, Hanting Su, Chunyuan Deng, Rui Feng, Yitong Li, Pranav Shetty, Aubrey Toland, Lingkai Kong, Changhao Li, Tianyi Chen, Isaac Rehg, Huan Tran, Lihua Chen, Will R. Gutekunst, Alexander J. Ratner, David Kartchner, Cassie Mitchell, Jiaming Shen, and Jimeng Sun. Our countless discussions, exchange of ideas, and collaborative spirit have enriched my work and contributed significantly to the completion of this dissertation.

Finally, I am grateful to Georgia Tech and to the outstanding faculty and staff in the Machine Learning program, School of ECE, CSE, and MSE. Their resources, support, and dedication to academic excellence have been instrumental in my personal and intellectual growth.

To everyone who has been part of this journey, mentioned here or not, I extend my deepest gratitude. This milestone would have been far more difficult to achieve without your generosity, and for that I am sincerely thankful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AUROC Area Under the Receiver Operating Characteristic
BBP Bayes by Backprop
BCE Binary Cross-Entropy
BERT Bidirectional Encoder Representations from Transformers
BNN Bayesian Neural Network
BS Brier Score
CHMM Conditional Hidden Markov Model
CoT Chain of Thought
CRF Conditional Random Field
CV Computer Vision
DNN Deep Neural Network
ECE Expected Calibration Error
ELMo Embeddings from Language Models
ELREA Ensembles of Low-Rank Expert Adapters
EM Expectation-Maximization
FC Fully Connected
G&O Generate and Organize
GCN Graph Convolutional Network
GIN Graph Isomorphism Network
GloVe Global Vectors
GNN Graph Neural Network
GPT Generative Pre-Trained Transformer
GRPO Group Relative Policy Optimization
HMM Hidden Markov Model
IE Information Extraction
KLD Kullback-Leibler Divergence
LF Labeling Function
LLM Large Language Model
LM Language Model
LoRA Low-Rank Adaptation
LSTM Long Short-Term Memory
MAE Mean Absolute Error
MC Dropout Monte Carlo Dropout
ML Machine Learning
MLM Masked Language Model
MoE Mixture of Experts

MoLE Mixture of LoRA Experts

MPNN Message Passing Neural Network

MRR Mean Reciprocal Rank

MSE Mean Squared Error

MUBen Molecular Uncertainty Benchmark

MV Majority Voting

NER Named Entity Recognition

NLL Negative Log-Likelihood

NLP Natural Language Processing

NN Neural Network

NTP Next-Token Prediction

OOD Out-Of-Distribution

PCA Principal Component Analysis

PEFT Parameter-Efficient Fine-Tuning

PLM Pre-Trained Language Model

RCE Regression Calibration Error

RE Relation Extraction

RL Reinforcement Learning

RLHF Reinforcement Learning from Human Feedback

RMSE Root Mean Square Error

SGD Stochastic Gradient Descent

SGLD Stochastic Gradient Langevin Dynamics

SMILES Simplified Molecular Input Line Entry System

SOTA State-of-the-Art

Sparse-CHMM Sparse Conditional Hidden Markov Model

SWAG Stochastic Weight Averaging-Gaussian

TS Temperature Scaling

UQ Uncertainty Quantification

UQAC Uncertainty Quantification with Attention Chain

WXOR Weighted XOR

SUMMARY

Pre-trained foundation models have become indispensable in modern Natural Language Processing (NLP) and scientific domains, evolving into Large Language Models (LLMs) with impressive zero- and few-shot capabilities. Despite their widespread success, challenges persist in real-world applications due to distribution shifts between training and inference data, opaque inference processes, and limited in-domain manually labeled examples. These issues complicate reliable confidence estimation and application of foundation models in downstream tasks. To address these concerns, this thesis explores two primary research directions: 1) reliable Uncertainty Quantification (UQ) and 2) data-efficient model learning.

In addressing reliability, we investigate different UQ methods and develop novel techniques to enhance model calibration. Molecular Uncertainty Benchmark (MUBen) establishes a best-practice benchmark for UQ in molecular representation models, thoroughly evaluating uncertainty calibration and predictive accuracy in large-scale discriminative tasks. Expanding uncertainty estimation techniques to autoregressive LLMs, we introduce Uncertainty Quantification with Attention Chain (UQAC), an approach that employs iterative attention-chain backtracking to approximate an otherwise intractable marginalization over Chain of Thought (CoT) reasoning paths, thus enhancing confidence estimation robustness for LLMs.

Regarding data efficiency, our work targets scenarios characterized by limited or noisy labeled data. In zero-shot Named Entity Recognition (NER), we develop Conditional Hidden Markov Model (CHMM) and Sparse Conditional Hidden Markov Model (Sparse-CHMM), which effectively exploit weak supervision signals through contextual embeddings from autoencoding foundation models, employing sparsity regularization to improve robustness. Additionally, we propose Generate and Organize (G&O), a zero-shot Information Extraction (IE) framework leveraging the powerful reasoning abilities of autoregressive LLMs. Lastly, we introduce Ensembles of Low-Rank Expert Adapters (ELREA), designed for date-efficient multi-task fine-tuning, which clusters training instructions based on gradient directions and applies task-specific Low-Rank Adaptation (LoRA) experts through ensemble techniques. ELREA mitigates task interference, promoting better generalization and parameter efficiency.

Together, our proposed methods enhance the trustworthiness and adaptability of pre-trained models in critical domains by addressing uncertainty concerns and reducing dependency on extensive labeled data. The thesis underscores the importance of calibration, interpretability, and scalable fine-tuning strategies in developing robust, data-efficient solutions suitable for high-stakes real-world applications.

CHAPTER 1
INTRODUCTION

1.1 Background

Recent years have witnessed significant advances in pre-trained foundation models. Building on the success of self-supervised word embedding methods such as Word2Vec [1] and Global Vectors (GloVe) [2] in the domain of NLP, the practice of training relatively simple Neural Networks (NNs) on large-scale datasets and transferring their learned representations to downstream tasks has been extended to other areas, including Computer Vision (CV) [3, 4], graph representation [5, 6, 7], and molecular embedding [8]. This paradigm has become a standard approach in Machine Learning (ML), as these embedding methods leverage large datasets to learn generalizable representations that can be used as fixed vectors for various downstream tasks.

The concept of self-supervised representation learning has led to the emergence of Pre-Trained Language Models (PLMs), exemplified by Embeddings from Language Models (ELMo) [9], Bidirectional Encoder Representations from Transformers (BERT) [10], and Generative Pre-Trained Transformer (GPT) [11]. ELMo, which employs Long Short-Term Memory (LSTM) networks [12], captures syntactic and semantic information via deep contextualized word representations and can be fine-tuned on downstream tasks by concatenating pre-trained weights with task-specific layers. Building on the Transformer architecture [13], BERT and GPT scale self-supervised objectives to unprecedented levels *w.r.t.* model parameters and training corpora, achieving remarkable success in NLP. They represent two main paradigms of PLMs: autoencoding and autoregressive, respectively. Autoencoding models such as BERT train Transformer encoders to reconstruct masked tokens using bidirectional context and excel in discriminative tasks such as sequence or token label prediction. Autoregressive models such as GPT use Transformer decoders to predict the next token based on previous tokens, a setup particularly effective for text generation tasks such as summarization, translation, and dialogue generation. Owing to the causal Next-Token Prediction (NTP) objective, GPT scales more effectively [14, 15] and generates high-quality text, inspiring LLMs such as GPT-3 [16], GPT-4 [17], Llama [18], Mistral [19], *etc.* These LLMs, often containing hundreds of billions of parameters, can perform many tasks in zero- or few-shot settings. Further enhancements, including CoT prompting [20], Mixture of Experts (MoE) [21], Reinforcement Learning from Human Feedback (RLHF) [22], LoRA [23], Group Relative Policy Optimization (GRPO) [24], and Reasoning-oriented Reinforcement Learning (RL) [25], bolster their capabilities in reasoning, coding, and mathematical problem solving.

The success of PLMs has likewise expanded to scientific domains such as chemistry and materials science, enabling breakthroughs in molecular property prediction [26, 27, 28, 29], protein structure prediction and design [30, 31, 32], and drug discovery [33]. Autoencoding models in these fields first treat molecules as token sequences (*e.g.*, Simplified Molecular Input Line Entry System (SMILES) strings [34]) and train with the Masked Language Model (MLM) objective [10] to reconstruct masked tokens, following approaches similar to those in NLP [27, 26, 35]. Subsequent works have explored alternative input representations such as 2D or 3D graphs, combined with training objectives like connection reconstruction or conformation prediction, to support foundational pre-training [36, 28, 29]. By contrast, autoregressive LLMs designed for scientific tasks often adapt general-purpose models via specialized prompting techniques [37] or conduct pre-training/fine-tuning on domain-specific *text* corpora [31, 32] to inject specialized knowledge. In the following discussions, we do not strictly distinguish between pre-trained foundation models and language models; instead, we uniformly refer to them as PLMs, despite potential variations in their input formats and training objectives.

1.2 Challenges

Despite the rapid progress in PLMs and their impressive capabilities, several critical challenges persist, particularly when these models are deployed in real-world scenarios. First, although PLMs exhibit remarkable performance in zero-shot or few-shot settings within domains aligned to their training data, they often struggle when presented with Out-Of-Distribution (OOD) data or with tasks that diverge significantly from their training distributions [20, 38]. Moreover, large-scale models, either in general or in scientific domains, can be prone to overfitting, frequently exhibiting misplaced confidence in their predictions [39, 40, 41]. This issue is exacerbated by the "black-box" nature of PLMs, which makes it difficult to ascertain how their outputs are derived or to precisely define the boundaries of their training distribution. Such opacity is particularly problematic in critical domains such as healthcare, finance, law, or materials design, where labeled evaluation data are scarce and the cost of errors can be substantial [42, 43, 44, 45, 46]. In these high-stakes contexts, ignoring uncertainty and reliability assessments may lead to erroneous decisions with serious consequences, including harm to individuals and significant financial losses. Hence, there is an urgent need to develop robust methods for interpretable UQ and reliability assessment of PLMs in such critical domains.

Another major challenge lies in data efficiency. While PLMs generally rely on large-scale pre-training, labeled data for fine-tuning can be limited or expensive to acquire in specialized areas [47, 48, 49]. Even in more general domains, substantial amounts of labeled data may still be required to achieve robust performance on certain downstream tasks, underscoring the continued importance of fine-tuning. This challenge becomes even more pronounced in scientific fields, where data annotation often demands extensive domain-specific expertise and is thus labor-intensive and costly [50, 51, 52, 53]. Consequently, there is a growing need for reliable data-efficient model learning and fine-tuning techniques that enable PLMs to be effectively adapted to specialized tasks despite scarce labeled resources. Developing such methods is vital to ensure PLMs can achieve their best performance in these resource-constrained, high-stakes scenarios.

1.3 Research Focus

To address the aforementioned challenges, this thesis pursues two main directions: 1) investigating and developing reliable methods for quantifying the uncertainty of PLMs; and 2) improving data-efficient approaches to tackle critical downstream tasks.

Reliable Uncertainty Quantification The first line of research begins with MUBen [41], a benchmarking framework designed to offer best practices for UQ in molecular property prediction (chapter 2). MUBen systematically compares a range of prevailing UQ techniques, drawn from multiple categories, and applies them to a variety of State-of-the-Art (SOTA) large-scale pre-trained discriminative molecular representation models that use different molecular descriptors [28, 36, 29]. Through rigorous evaluations of prediction and calibration performance across multiple tasks, MUBen distills actionable insights on how to build more reliable, uncertainty-aware molecular models.

Continuing on this front, we propose a novel UQ pipeline for autoregressive LLMs, named UQAC [54] (chapter 3). Advanced autoregressive LLMs generate CoT reasoning steps in their responses for better accuracy; but UQAC focuses on measuring the model's confidence only in its *final* answer, effectively requiring an intractable marginalization over all possible reasoning paths. To make the marginalization practical, UQAC iteratively constructs an "attention chain" of tokens deemed crucial to the final output via a specialized backtracking procedure. By integrating this attention chain with probability-based candidate-token thresholding, UQAC significantly reduces the space of plausible reasoning paths and consequently offers a more reliable confidence estimate of the final answer.

Data-Efficient Model Learning The second branch of this thesis tackles data efficiency in specialized settings. We first investigate the zero-shot NER problem, *i.e.*, extracting structured named entities from text without any manual annotations (chapter 4). The primary focus is on weak supervision approaches, which rely on multiple weak Labeling Functions (LFs) to generate diverse but noisy labels for each instance and then integrate them into a uniformed sequence [55, 56]. Building upon Hidden Markov Models (HMMs) for label aggregation [57, 58], we introduce CHMM [59], which leverages contextual embeddings from autoencoding PLMs such as BERT to incorporate richer input semantics. Its successor, Sparse-CHMM [60], imposes sparsity constraints on emission matrices, improving both training efficiency and robustness in cases where the numbers of labels or LFs are large. With the rapid advancements in LLMs, the thesis then presents G&O [61], which explores effective strategies to harness powerful autoregressive LLMs for zero-shot IE tasks.

The thesis then addresses the challenge of data-efficient training and fine-tuning for autoregressive LLMs, particularly when data originate from diverse sources. Recent findings suggest that large-scale fine-tuning can suffer from conflicting gradient directions across tasks, sometimes harming model generalization. Further, research indicates that carefully curating smaller task-specific datasets can match or surpass the performance of fine-tuning with all available data [62]. Consequently, we introduce ELREA [63] (chapter 5), a method designed to maximize training-data utilization while mitigating conflicting gradient updates. ELREA clusters training instructions by their gradient directions, effectively partitioning the data into distinct "expertise regions." Expert adapters are then trained on these clusters using LoRA, enhancing both efficiency and model scalability. During inference, ELREA automatically identifies which cluster and the corresponding adapter is most relevant for a given input, based on the similarity of gradient directions. In doing so, ELREA ensures that each input is handled by the most suitable adapter, thereby improving generalization and performance on diverse downstream tasks.

1.4 Broader Impact

The contributions in this thesis have broader implications both within and beyond traditional ML research. First, improving UQ in PLMs holds significant societal importance, particularly for sensitive fields such as healthcare, finance, and law, where overly confident or poorly calibrated predictions can result in grave consequences. By providing robust and interpretable approaches to UQ, we empower practitioners and researchers to better assess and mitigate the risk of model mispredictions, thereby enhancing decision-making processes in high-stakes settings.

Second, advancements in data efficiency can substantially broaden the accessibility of PLMs across various domains, especially those with limited labeled data. Rather than requiring massive annotated datasets for each new task, data-efficient approaches enable practitioners to make effective use of smaller, carefully curated datasets or weak supervision signals. This not only lowers the barrier to entry for small organizations or research teams with limited annotation budgets but also reduces the computational overhead, promoting more sustainable and environmentally conscious AI practices.

Finally, the methodological innovations in this thesis span uncertainty benchmarking, autoregressive confidence estimation, and specialized fine-tuning, underscoring the importance of open and reproducible science. The proposed frameworks and pipelines are designed to be generalizable across different application domains, and we strive to release code, data, and results to the research community. By doing so, we aim to facilitate further development, foster collaborative research, and promote responsible deployment of PLMs in real-world scenarios. Collectively, these efforts help ensure that high-performing language models remain transparent, reliable, and accessible, ultimately contributing to the ethical and equitable advancement of AI technologies.

CHAPTER 2
BENCHMARKING MOLECULAR REPRESENTATION UNCERTAINTIES

2.1 Introduction

Learning effective molecular, protain, or material representations is critical for satisfying target prediction performance in a wide range of scientific tasks across chemistry, biology, and materials science [64]. With the emergence of self-supervised PLMs, there has been a surge of interest in leveraging vast unlabeled molecular datasets to build large-scale pre-trained molecular representation models (referred to as molecular PLMs herein) [27, 65, 36, 28, 29, 66]. These models have demonstrated impressive representational power, achieving SOTA performance in various molecular property prediction tasks [67].

Molecular PLMs can be broadly characterized by the descriptor types they employ. Earlier works often utilize string-based inputs (*e.g.*, ChemBERTa [26]) or 2D graph-based methods (*e.g.*, Graph Isomorphism Network (GIN) and GROVER [65, 68]), each capturing distinct structural and chemical cues. In parallel, explicit 3D geometries have recently gained traction for their ability to capture richer spatial information. Among these 3D-based approaches, invariant models (*e.g.*, SchNet [69], SphereNet [70], GemNet [71], *etc.*) leverage scalar features that remain unchanged under rotations, translations, and reflections, while equivariant models (*e.g.*, TFN [72], Equiformer [73], LEFTNet [74], *etc.*) maintain consistent representations when subjected to these transformations. Although 3D modeling typically demands more computational resources and geometric information, it has shown promise in boosting predictive accuracy, particularly for geometrically sensitive tasks.

While predictive performance remains central to molecular modeling, real-world applications often require reliable estimates of model confidence to mitigate risk. Accurate UQ facilitates the identification of uncertain or unreliable predictions, which is increasingly critical for tasks such as high-throughput screening [75, 76], detecting activity cliffs [77], and guiding experimental design [43, 44]. However, large-scale PLMs frequently overfit during fine-tuning, yielding overconfident yet erroneous predictions [39].

A variety of UQ strategies have been recently investigated [78], encompassing both Bayesian methods (*e.g.*, Monte Carlo Dropout (MC Dropout) [79]) and non-Bayesian approaches (*e.g.*, Deep Ensembles [80], evidential networks [81], and post-hoc calibration [82]). For example, [83] incorporate post-hoc recalibration [39] into a Message Passing Neural Network (MPNN) [84] to mitigate overconfidence in molecular property prediction; [44] apply evidential message passing networks [81, 85] to quantitative structure-activity relationship regression; and [86] use Bayesian optimization for reliable predictions of nanoporous material properties and experimental guidance.

Despite these contributions, several important limitations persist. 1) The variety of uncertainty estimation methods considered is often narrow, overlooking potentially effective strategies. 2) Most existing research focuses on a confined set of molecular properties (often quantum-mechanical), rather than exploring a broader spectrum of tasks. 3) To date, none of these investigations fully embrace recent pre-trained molecular backbones, whose strong predictive performance and unique architectural characteristics could distinctly affect UQ outcomes. Accordingly, a comprehensive study on how different UQ methods perform when paired with modern pre-trained molecular PLMs is still lacking and remains an urgent research gap.

To bridge these gaps, we publish MUBen [41], a benchmark specifically designed to systematically evaluate UQ methods across diverse molecular PLMs and multiple property prediction tasks, as illustrated in Figure 2.1. MUBen explores UQ methods spanning deterministic prediction, Bayesian Neural Networks (BNNs), post-hoc calibration, and Deep Ensembles, deploying them on a broad set of molecular PLMs with different descriptor types (*e.g.*, string, 2D, and 3D graphs) and architectures (*e.g.*, Deep Neural Network (DNN), Graph Neural Network (GNN), and Transformers). Through extensive experiments and

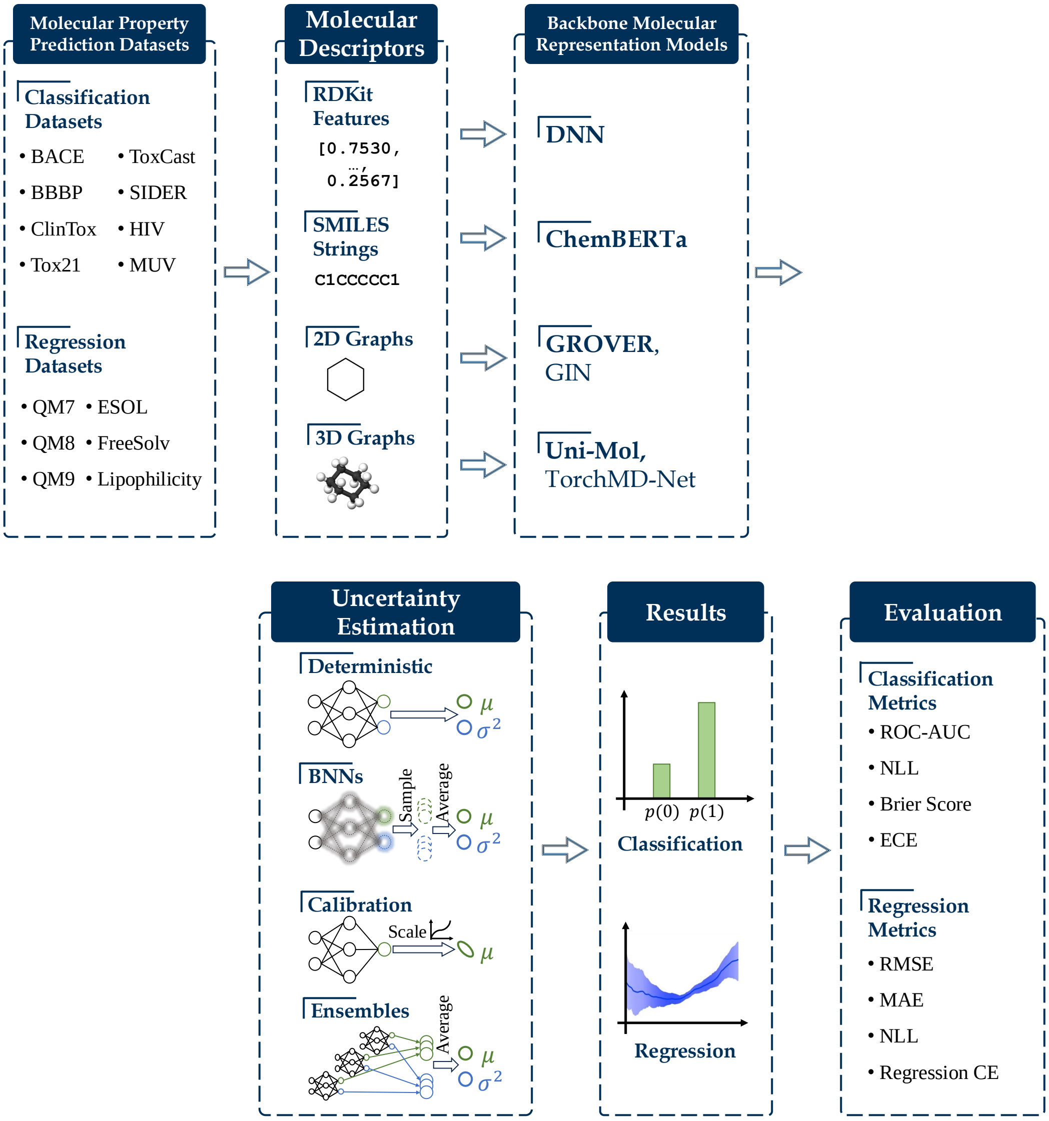


Figure 2.1: MUBen overview.

analyses, MUBen aims to yield new insights and best practices for model and UQ strategy selection in practical molecular modeling workflows. Our open-source implementation is available at https://github.com/Yinghao-Li/MUBen, featuring a modular design that is user-friendly, transferable, and straightforward to extend. We hope that MUBen will stimulate the future development of UQ methods, pre-trained molecular models, and their applications in materials science and drug discovery.

2.2 Problem Setup

Let $\boldsymbol{x}$ denote a descriptor of a molecule, such as a SMILES string or a 2D topological graph. Variable y is the label with domain $y \in \{1, \ldots, K\}$ for K-class classification or $y \in \mathbb{R}$ for regression. $\boldsymbol{\theta}$ denotes all parameters of the molecular PLM $\mathcal{M}$ and the additional task-specific output layer. We assume that a training dataset $\mathbb{D}$ consists of N *i.i.d.* samples $\mathbb{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$. The goal of an uncertainty-aware molecular PLM fine-tuning is to optimize the model parameters $\boldsymbol{\theta}$ on $\mathbb{D}$ in order to develop a predictive model that accurately estimates the probability distribution $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n)$. To estimate how well the model captures the uncertainty of the predictions, we introduce the following metrics to evaluate the UQ performance from different perspectives.

2.2.1 Negative Log-Likelihood

Negative Log-Likelihood (NLL) is often utilized to assess the quality of model uncertainty on holdout sets for both classification and regression tasks. Despite being a proper scoring rule as per Gneiting's framework [87], certain limitations such as overemphasizing tail probabilities [88] make it inadequate to serve as the only UQ metric. For binary classification with Sigmoid output activation, NLL is given by:

$$\text{NLL} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{p}_n + (1 - y_n)\log(1 - \hat{p}_n)\right], \tag{2.1}$$

where $y_n \in \{0, 1\}$ is true label and $\hat{p}_n \in (0, 1) = p_{\boldsymbol{\theta}}(\hat{y}_n \mid \boldsymbol{x}_n)$ is predicted probability. For regression, the Gaussian NLL is calculated as:

$$\text{NLL} = -\frac{1}{N}\sum_{n=1}^{N}\log\mathcal{N}(y_n; \hat{y}_n, \hat{\sigma}_n) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left[\log(2\pi\hat{\sigma}_n) + \frac{(y_n - \hat{y}_n)^2}{\hat{\sigma}_n}\right], \tag{2.2}$$

where $\hat{y}_n \in \mathbb{R}$ is the predicted mean and $\hat{\sigma}_n \in \mathbb{R}_+$ is the predicted variance regularized by the SoftPlus activation.

2.2.2 Brier Score

Brier Score (BS) [89] quantifies the mean squared error between predicted probabilities and actual labels, and is recognized as a proper scoring rule for classification tasks. The formula for the BS is given by:

$$\mathrm{BS} = \frac{1}{N} \sum_{n=1}^{N} \sum_{y=1}^{K} \left(p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_n) - \mathbb{I}(y = y_n)\right)^2 , \tag{2.3}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Specifically, $\mathbb{I}(a = b) = 1$ if $a = b$; otherwise, $\mathbb{I}(a = b) = 0$.

2.2.3 Calibration Errors

Calibration Errors measure the correspondence between predicted probabilities and empirical accuracy. For classification, we use Expected Calibration Error (ECE) [90], which first divides the predicted probabilities $\{p_{\boldsymbol{\theta}}(y_n \mid \boldsymbol{x}_n)\}_{n=1}^{N}$ into S bins $\mathbb{B}_s = \{n \in \{1, \dots, N\} \mid p_{\boldsymbol{\theta}}(y_n \mid \boldsymbol{x}_n) \in (\rho_s, \rho_{s+1}]\}$ with $s \in \{1, \dots, S\}$, each with size N_s, and then compute the L_1 loss between the accuracy and predicted probabilities within each bin:

$$\mathrm{ECE} = \sum_{s=1}^{S} \frac{N_s}{N} \left| \frac{1}{N_s} \sum_{n \in \mathbb{B}_s} \mathbb{I}(y_n = \hat{y}_n) - \frac{1}{N_s} \sum_{n \in \mathbb{B}_s} p_{\boldsymbol{\theta}}(\hat{y}_n \mid \boldsymbol{x}_n) \right| , \tag{2.4}$$

where $\hat{y}_n = \arg\max_y p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}_n)$ is the n-th prediction.

For regression tasks, Regression Calibration Error (RCE) [91] measures the accuracy of prediction intervals. It calculates the true frequency of the predicted points falling within each confidence interval against the predicted fraction of points in that interval:

$$\mathrm{CE} = \frac{1}{S} \sum_{s=1}^{S} \left(\frac{s}{S} - \frac{1}{N} \left| \{n \in \{1, \dots, N\} \mid F_{\boldsymbol{\theta}}(y_n; \boldsymbol{x}_n) \leq \frac{s}{S}\} \right| \right)^2 , \tag{2.5}$$

where $\frac{s}{S}$ represents the expected quantile. $F_{\boldsymbol{\theta}}(y_n; \boldsymbol{x}_n)$ denotes the predicted quantile value

Table 2.1: Model statistics.

Model	# Parameters (M)	Average Time per Training Step (ms)[(a)]
DNN	0.158	5.39
ChemBERTa	3.43	30.18
GROVER	48.71	334.47
Uni-Mol	47.59	392.55
TorchMD-NET	7.23	217.29
GIN	0.26	7.21

[(a)] All models are evaluated on the BBBP dataset with a batch size of 128. We only measure the time for forward passing, backward passing, and parameter updating. We train the model for 6 epochs and take the average of the last 5 to reduce the impact of GPU initialization.

for y_n, such as a Gaussian cumulative distribution function $\Phi(y_n; \hat{y}_n, \hat{\sigma}_n)$, which is parameterized by the estimated mean $\hat{y}_n$ and variance $\hat{\sigma}_n$ assuming a Gaussian distribution.

2.3 Experiment Setup

2.3.1 Molecular Descriptors and Backbone Models

Molecular descriptors transform molecular structures into computational representations—such as fingerprints, SMILES strings, or graphs—thereby enabling various machine learning and computational analyses. Different descriptors, when paired with model architectures that have distinct inductive biases, can offer complementary advantages in downstream tasks.

In this work, we select 4 primary backbone models, each distinguished by its input descriptor: **1)** ChemBERTa [26, 35], a Transformer-based architecture that processes SMILES strings; **2)** GROVER [36], which combines Transformer encoders and dynamic message-passing GNNs for 2D molecular graphs; **3)** Uni-Mol [29], a Transformer specializing in 3D molecular conformations to capture spatial relationships; **4)** a fully connected DNN with fixed 200-dimensional RDKit features, serving as a baseline to compare heuristic versus learned representations.

Additionally, we incorporate two supplementary backbones for broader exploration: **5)** TorchMD-NET [92, 93], an equivariant Transformer architecture pre-trained on quan-

Table 2.2: Computation resources required by different uncertainty estimation methods. We assume that we have already trained a deterministic backbone model for property prediction, and would like to build up a UQ method on top of it.

UQ Method	Training Starting Checkpoint	Additional Cost[(a)]
Deterministic	-	0
Temperature	from fine-tuned backbone	$(T_{\text{infer}} + T_{\text{train-FFN}}) \times M_{\text{train-extra}}$
Focal Loss	from scratch	$T_{\text{train}} \times M_{\text{train}}$
MC Dropout	no training	$T_{\text{infer}} \times M_{\text{infer}}$
SWAG	from fine-tuned backbone	$T_{\text{train}} \times M_{\text{train-extra}} + T_{\text{infer}} \times M_{\text{infer}}$
BBP	from scratch	$T_{\text{train}} \times M_{\text{train}} + T_{\text{infer}} \times M_{\text{infer}}$
SGLD	from scratch	$T_{\text{train}} \times (M_{\text{train}} + M_{\text{train-extra}}) + T_{\text{infer}} \times M_{\text{infer}}$
Ensembles	from scratch	$T_{\text{train}} \times M_{\text{train}} \times (N_{\text{ensembles}} - 1)$

[(a)] T_{train} and T_{infer} are the time for one epoch of training and inference of the backbone model, respectively. In general, $T_{\text{train}} \gg T_{\text{infer}}$. M_{train}, $M_{\text{train-extra}}$, and M_{infer} are the number of training epochs, additional training epochs, and inference epochs, respectively (section A.3.1). In general, $M_{\text{train}} \gg M_{\text{train-extra}}$. Different backbones and UQ methods have different Ts and Ms, but we use the same symbols nonetheless for simplicity. The result is a rough estimation without considering the additional inference time or the early stopping if a model is retrained.

tum mechanical data; **6)** GIN [94], a randomly initialized GNN baseline operating on 2D graphs, included as a simpler point of comparison without any pre-trained features.

A summary of model statistics is presented in Table 2.1, and detailed model descriptions can be found in section A.2.

2.3.2 Uncertainty Quantification Methods

In MUBen, we examine several popular UQ methods that span different methodological families, as described below.

Deterministic Uncertainty Prediction A common approach for binary classification tasks is to use Sigmoid outputs, interpreting the predicted class probabilities as measures of uncertainty. For regression tasks, networks often predict both the mean and variance of an independent Gaussian distribution, with the variance acting as an uncertainty estimate. We label this approach as “deterministic.”

Focal Loss [95, 96] offers an alternative loss function that minimizes a regularized

Kullback-Leibler Divergence (KLD) between predicted logits and true labels. The added regularization term increases the entropy of the predicted distribution, reducing overconfidence and potentially improving uncertainty calibration.

Bayesian Learning and Inference BNNs approximate a posterior distribution over network parameters, capturing prediction uncertainty via random draws from these distributions:

- Bayes by Backprop (BBP) [97] introduces Monte Carlo gradients, extending the Gaussian reparameterization trick [98] to directly learn the posterior distribution of weights through backpropagation.
- Stochastic Gradient Langevin Dynamics (SGLD) [99] injects noise into the Stochastic Gradient Descent (SGD) process via Langevin dynamics, thereby generating samples that can be used for Monte Carlo estimates of posterior expectations.
- MC Dropout [79] interprets dropout as a Bayesian approximation to a deep Gaussian process [100]. Multiple stochastic forward passes (with dropout enabled) yield a distribution of predictions reflecting model uncertainty.
- Stochastic Weight Averaging-Gaussian (SWAG) [101] leverages stochastic weight averaging [102] to estimate Gaussian posteriors with low-rank and diagonal approximations of network weight covariance.

Post-Hoc Calibration Post-hoc calibration tackles the overconfidence issue of deterministic models by adjusting their output distributions after training. The most widely used approach is Temperature Scaling (TS) [82, 39], which introduces a learned scaling factor to modulate the “sharpness” of SoftMax or Sigmoid activations, enhancing model calibration.

Deep Ensembles Ensembles have long been used to boost performance in machine learning [103] and were first adopted for uncertainty estimation by [80]. This method trains

Table 2.3: Dataset statistics.

Property Category	Dataset	# Compounds	# Tasks	Average LIR$^{(a)}$	Max LIR$^{(a)}$
		Classification			
Physiology	BBBP	2,039	1	0.7651	0.7651
	ClinTox	1,478	2	0.9303	0.9364
	Tox21	7,831	12	0.9225	0.9712
	ToxCast	8,575	617	0.8336	0.9972
	SIDER	1,427	27	0.7485	0.9846
Biophysics	BACE	1,513	1	0.5433	0.5433
	HIV	41,127	1	0.9649	0.9649
	MUV	93,087	17	0.9980	0.9984
		Regression			
Physical Chemistry	ESOL	1,128	1	-	-
	FreeSolv	642	1	-	-
	Lipophilicity	4,200	1	-	-
Quantum Mechanics	QM7	7,160	1	-	-
	QM8	21,786	12	-	-
	QM9	133,885	3$^{(b)}$	-	-

$^{(a)}$ LIR stands for "label imbalance ratio": $\text{LIR}_k \in [0.5, 1] = \max\{n_{\text{pos}}, n_{\text{neg}}\}/n_{\text{total}}$.
$^{(b)}$ QM9 dataset contains 12 tasks, but we follow [105, 29] and use only 3 most popular ones (homo, lumo, and gap).

multiple deterministic models with distinct random seeds and aggregates their predictions. By exploring various modes in the loss landscape, ensembles are more resilient to noise and OOD instances [104].

Table 2.2 summarizes the computational overhead of each UQ method. Additional details are available in section A.3.

2.3.3 Datasets

We validate the above approaches on MoleculeNet [67], a popular suite of molecular property datasets spanning quantum mechanics, solubility, toxicity, and more. For classification, MUBen incorporates BBBP, ClinTox, Tox21, ToxCast, SIDER, BACE, HIV, and MUV, all of which are binary-label tasks. The first 5 datasets assess **physiological** properties, while the last 3 focus on **biophysical** endpoints. For regression, MUBen includes

ESOL, FreeSolv, Lipophilicity, QM7, QM8, and QM9. The first 3 represent **physical chemistry** properties, while the latter 3 examine **quantum mechanics** targets.

In line with earlier studies [105, 29], we adopt scaffold splitting to better emulate real-world generalization and reduce the influence of dataset randomness. Scaffold splitting forces each subset to contain distinct scaffolds, creating a more realistic OOD scenario. For completeness, we also report results using random splitting. Further dataset information is provided in section A.1.

2.3.4 Training and Evaluation Protocols

For classification, we use Sigmoid activation with a Binary Cross-Entropy (BCE) loss unless otherwise specified. For regression, targets are standardized to a Gaussian distribution during training and converted back for final evaluation. Uncertainty is modeled by predicting a mean and variance for each data point, with a SoftPlus activation ensuring positive variance estimates.

Our primary performance metrics are Area Under the Receiver Operating Characteristic (AUROC) for classification, and Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) for regression. AUROC is widely used for binary classification. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various decision thresholds $t \in (0, 1)$. Given a set of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), the TPR and FPR are computed as $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$ and $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$ respectively. The AUC signifies the likelihood of a randomly selected positive instance being ranked above a randomly chosen negative instance. This integral under the ROC curve is calculated as

$$\text{AUROC} = \int_0^1 \text{TPR}(t) \frac{d}{dt} \text{FPR}(t) dt \tag{2.6}$$

and can be approximated using numerical methods.

For regression tasks, RMSE quantifies the average discrepancy between predicted values $\hat{y}_n \in \mathbb{R}$ and actual values $y_n \in \mathbb{R}$ for N data points, given by:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}. \tag{2.7}$$

MAE is another regression metric, measuring the average absolute deviation between predicted and actual values. It is calculated as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_n - y_n|. \tag{2.8}$$

Uncertainty is evaluated using ECE, NLL, and BS in classification tasks; and Gaussian NLL and RCE for regression, as described in section 2.2. Unless stated otherwise, we report results averaged across three independent training runs, and for Deep Ensembles we fuse predictions before metric computation. These protocols align with established benchmarks [67, 105, 29].

2.4 Experiments and Analysis

2.4.1 Prediction Performance

Theoretically, inserting UQ methods into the training pipeline does not guarantee better prediction on i.i.d. datasets. However, since we ensure OOD test points with scaffold splitting, UQ methods may mitigate the distribution gap, yielding better test results. The columns AUROC, RMSE, and MAE in Table 2.4, Table 2.5, and Figure 2.2 illustrate the predictive performance of each method. From the lens of UQ methods (Figure 2.3), the randomness in the initialization and training trajectory of Deep Ensembles explores a broader range of loss landscapes, which partially addresses the distribution shift issue. MC Dropout may flatten extreme regression abnormality triggered by OOD features. This phenomenon is less pronounced for classification due to the $(0, 1)$ output domain. However, other BNNs

Table 2.4: Classification results. "↑" and "↓" imply that better performance is indicated by a larger or smaller value, respectively. Text in bold signifies the top-performing UQ method within each backbone model, and cells in blue indicate the best performance across all backbone-UQ combinations. "ROC" represents ROC-AUC. Deep Ensembles consistently outperforms other UQ methods for both property prediction and uncertainty quantification; MC Dropout and Temperature Scaling also consistently improve UQ performance.

	Tox21[a]				**ToxCast**[a]				**Average Ranking**[b]			
	ROC↑	ECE↓	NLL↓	BS↓	ROC↑	ECE↓	NLL↓	BS↓	ROC↓	ECE↓	NLL↓	BS↓
DNN-RDKit												
Deterministic	0.7386	0.0417	0.2771	0.0779	0.6222	0.1168	0.4436	0.1397	25.75	17.25	24.13	22.88
Temperature	0.7386	**0.0342**	0.2723	0.0773	0.6220	0.1114	0.4882	0.1398	25.75	15.38	21.25	19.88
Focal Loss	0.7374	0.1058	0.3161	0.0871	0.6289	0.1264	0.4389	0.1396	25.88	24.38	22.38	24.00
MC Dropout	0.7376	0.0356	0.2727	0.0763	0.6248	0.1093	0.4319	0.1358	26.50	13.00	19.38	18.63
SWAG	0.7364	0.0438	0.2793	0.0790	0.6207	0.1175	0.4441	0.1400	26.38	18.50	25.63	23.50
BBP	0.7243	0.0422	0.2847	0.0814	0.6020	0.1443	0.4673	0.1510	22.75	19.13	19.38	21.38
SGLD	0.7257	0.1192	0.3455	0.0978	0.5319	0.3054	0.6685	0.2378	27.75	26.00	28.88	27.88
Ensembles	**0.7540**	0.0344	**0.2648**	**0.0746**	**0.6486**	**0.0900**	**0.4008**	**0.1292**	**20.00**	**7.13**	**11.75**	**13.13**
ChemBERTa												
Deterministic	0.7542	0.0571	0.2962	0.0812	0.6554	0.1209	0.4313	0.1330	15.63	17.38	18.88	19.38
Temperature	0.7542	0.0424	0.2744	0.0792	0.6540	0.1067	0.4817	0.1313	15.88	12.00	13.88	15.38
Focal Loss	0.7523	0.0969	0.3052	0.0845	0.6442	0.1197	0.4243	0.1346	17.63	20.13	17.88	20.63
MC Dropout	0.7641	0.0423	0.2697	**0.0744**	0.6624	0.1069	0.4070	0.1276	12.50	**10.75**	**10.63**	**10.00**
SWAG	0.7538	0.0592	0.3008	0.0818	0.6556	0.1202	0.4305	0.1327	16.13	19.50	21.38	20.38
BBP	0.7433	0.0459	0.2765	0.0780	0.5814	0.1276	0.4545	0.1469	20.88	19.00	16.00	19.50
SGLD	0.7475	0.0504	0.2784	0.0795	0.5436	0.2238	0.5602	0.1881	21.13	19.88	19.13	18.63
Ensembles	**0.7681**	**0.0440**	**0.2679**	0.0750	**0.6733**	**0.1037**	**0.3986**	**0.1258**	**12.38**	13.00	11.63	12.25
GROVER												
Deterministic	0.7808	0.0358	0.2473	0.0694	0.6587	0.1043	0.4091	0.1298	11.63	11.50	8.88	9.88
Temperature	0.7810	**0.0291**	0.2439	0.0686	0.6496	0.1424	0.4612	0.1424	12.63	8.88	7.50	9.25
Focal Loss	0.7779	0.1148	0.3052	0.0811	0.6359	0.1221	0.4365	0.1383	15.00	23.38	21.50	22.25
MC Dropout	0.7817	0.0346	0.2455	0.0689	0.6615	**0.1009**	**0.4042**	**0.1288**	11.25	10.50	7.63	8.63
SWAG	0.7837	0.0360	0.2482	0.0689	0.6603	0.1060	0.4114	0.1301	9.13	11.63	8.88	8.38
BBP	0.7697	0.0438	0.2552	0.0711	0.5995	0.1731	0.5090	0.1660	16.75	22.00	14.75	15.88
SGLD	0.7635	0.0402	0.2558	0.0716	0.5542	0.2712	0.6194	0.2139	18.75	18.63	15.25	15.63
Ensembles	**0.7876**	0.0316	**0.2411**	**0.0675**	**0.6646**	0.1034	0.4061	0.1290	**8.50**	**8.13**	**5.38**	**6.88**
Uni-Mol												
Deterministic	0.7895	0.0454	0.2601	0.0716	0.6734	0.1020	0.3983	0.1274	11.50	15.50	16.13	13.88
Temperature	0.7896	0.0346	0.2483	0.0704	**0.7028**	0.1456	0.4566	0.1355	11.00	12.00	13.13	12.75
Focal Loss	0.7904	0.0972	0.2899	0.0785	0.6934	0.1227	0.4079	0.1284	10.00	23.50	19.38	20.50
MC Dropout	0.7891	0.0480	0.2628	0.0726	0.6833	0.1074	0.4015	0.1274	12.88	19.88	19.25	17.25
SWAG	0.7842	0.0593	0.2994	0.0728	0.6870	0.1085	0.4005	0.1271	10.13	22.13	22.25	18.25
BBP	0.7932	0.0396	0.2520	0.0703	0.6273	0.1296	0.4522	0.1456	12.13	17.00	15.50	16.25
SGLD	0.7887	0.0433	0.2569	0.0684	0.5700	0.1953	0.5207	0.1717	14.75	18.13	18.88	14.50
Ensembles	**0.8052**	**0.0332**	**0.2389**	**0.0662**	0.6841	**0.0953**	**0.3877**	**0.1247**	**5.13**	**8.88**	**7.63**	**6.50**
TorchMD-NET[c]	0.7793	0.0409	0.2614	0.0708	0.6540	0.1546	0.4424	0.1396	-	-	-	-
GIN[c]	0.6829	0.0634	0.3268	0.0840	0.5752	0.1381	0.4835	0.1477	-	-	-	-

[a] The "Tox21" and "ToxCast" columns present metric scores on representative exemplar datasets, highlighting the trends observable across all datasets.

[b] The "Average Ranking" columns provide the rank of each model's UQ metrics against all other backbone-UQ combinations averaged from all classification datasets; smaller number indicates better performance.

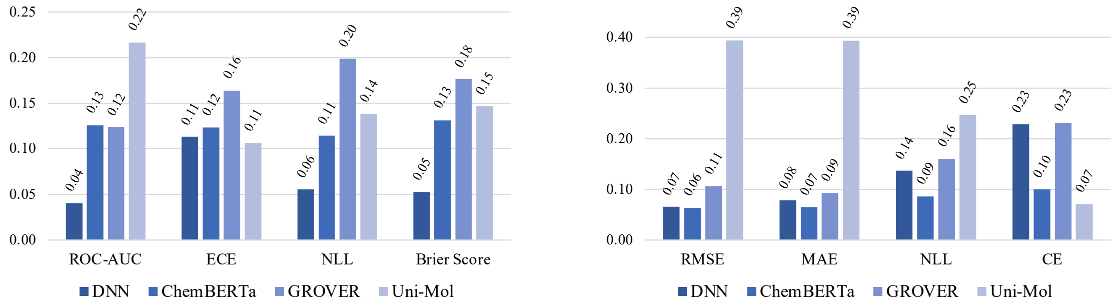
[c] We report the results from the best-performing UQ method—Deep Ensembles for TorchMD-NET and GIN. These backbones are not ranked together with the primary benchmark.

Table 2.5: The regression outcomes for property prediction and uncertainty quantification, where lower scores are preferable, cover two example datasets along with the average ranking. Similar to classification, Deep Ensembles consistently outperforms other methods. Specifically within the context of regression, BBP and SGLD demonstrate exceptional capabilities in estimating uncertainty, despite not consistently improving property prediction outcomes.

	Lipophilicity				**QM9**				**Average Ranking**			
	RMSE	MAE	NLL	CE	RMSE	MAE	NLL	CE	RMSE	MAE	NLL	CE
						DNN-RDKit						
Deterministic	0.7575	0.5793	0.6154	0.0293	0.01511	0.01012	-3.379	0.04419	16.67	15.67	11.33	10.50
MC Dropout	0.7559	0.5773	0.9071	0.0341	0.01480	0.01000	-3.526	0.04327	15.17	15.17	11.33	10.67
SWAG	0.7572	0.5823	0.7191	0.0308	0.01524	0.01019	-3.284	0.04495	18.00	17.67	14.00	12.50
BBP	0.7730	0.5938	0.7578	0.0305	0.01534	0.01025	-3.347	0.04452	21.17	20.50	10.33	7.33
SGLD	0.7468	0.5743	**0.2152**	**0.0090**	0.01958	0.01437	-3.335	**0.00702**	19.83	19.33	6.83	**1.83**
Ensembles	**0.7172**	**0.5490**	0.6165	0.0322	**0.01430**	**0.00956**	**-3.602**	0.04362	**11.50**	**11.17**	**6.33**	8.67
						ChemBERTa						
Deterministic	0.7553	0.5910	1.2368	0.0362	0.01464	0.00916	-2.410	0.05468	17.83	16.67	18.83	14.67
MC Dropout	**0.7142**	**0.5601**	0.8178	0.0349	0.01412	0.00880	-3.150	0.05133	**14.33**	**13.83**	15.00	13.67
SWAG	0.7672	0.5992	1.5809	0.0395	0.01477	0.00925	-2.170	0.05535	19.33	18.50	20.33	15.83
BBP	0.7542	0.5869	**0.4419**	**0.0279**	0.01443	0.00928	-2.593	0.05399	17.67	18.50	10.83	9.00
SGLD	0.7622	0.5982	0.8719	0.0355	0.01530	0.01012	**-3.758**	**0.03378**	19.83	20.50	**9.50**	**8.50**
Ensembles	0.7367	0.5763	0.9756	0.0360	**0.01397**	**0.00868**	-2.876	0.05425	14.83	**13.83**	14.67	12.67
						GROVER						
Deterministic	0.6316	0.4747	2.1512	0.0478	0.01148	0.00678	-0.787	0.06206	10.67	11.67	17.67	16.00
MC Dropout	0.6293	0.4740	2.0526	0.0476	0.01140	0.00676	-1.100	0.06161	9.33	11.17	16.00	14.33
SWAG	0.6317	0.4750	2.3980	0.0485	0.01156	0.00678	-0.477	0.06252	12.33	13.33	20.33	17.83
BBP	0.6481	0.5058	0.0789	0.0196	0.01179	0.00700	-1.885	0.05909	14.17	15.67	7.67	4.00
SGLD	0.6360	0.4984	**0.0544**	**0.0215**	0.01359	0.00878	**-3.785**	**0.02911**	14.83	15.17	**4.67**	**2.67**
Ensembles	**0.6250**	**0.4693**	1.6046	0.0460	**0.01143**	**0.00667**	-1.028	0.06199	**8.17**	**8.67**	13.50	14.00
						Uni-Mol						
Deterministic	0.6079	0.4509	0.8975	0.0425	0.00962	0.00538	0.014	0.06637	5.83	4.83	16.67	21.17
MC Dropout	0.5983	0.4438	1.3663	0.0440	0.00961	0.00535	-0.251	0.06615	4.00	3.17	16.67	21.33
SWAG	0.6026	0.4476	1.0101	0.0453	0.00969	0.00541	-0.462	0.06597	6.33	6.17	19.67	22.67
BBP	0.6044	0.4469	**0.0679**	**0.0306**	0.00952	0.00544	-2.959	0.06179	3.17	3.00	4.33	10.33
SGLD	0.6040	0.4554	0.1565	0.0329	0.00950	0.00546	**-4.209**	**0.04593**	**2.50**	4.00	**2.50**	**9.17**
Ensembles	**0.5809**	**0.4266**	0.6450	0.0438	**0.00948**	**0.00526**	-0.319	0.06629	**2.50**	**1.83**	11.00	20.67
TorchMD-NET[a]	1.0313	0.8196	0.8619	0.0195	0.00860	0.00464	2.262	0.06868	-	-	-	-
GIN[a]	0.8071	0.6515	0.3241	0.0020	0.01295	0.00814	-3.521	0.04997	-	-	-	-

[a] We report the results from the Deep Ensembles UQ method for TorchMD-NET and GIN.

do not exhibit the same advantage. SWAG, while similar to MC Dropout *w.r.t.* training, might intensify training data overfitting due to the additional steps taken to fit the parameter distribution. The stochastic sampling employed by BBP and SGLD complicates network training and may impact the prediction performance as a result. Looking from the perspective of primary backbones (Figure 2.2), Uni-Mol secures the best prediction performance for both classification and regression. The superior molecular representation capability of Uni-Mol is attributed to the large network size, the various pre-training data and tasks, and the integration of results from different conformations of the same molecule. When contrasting DNN, ChemBERTa, and GROVER, it becomes pronounced that the expressiveness of the molecular descriptors varies for different molecules/tasks. Moreover, pre-trained models do not invariably surpass heuristic features when integrated with UQ methods. Selecting a model attuned to the specific task is advised over an indiscriminate reliance on "deep and complex" networks.



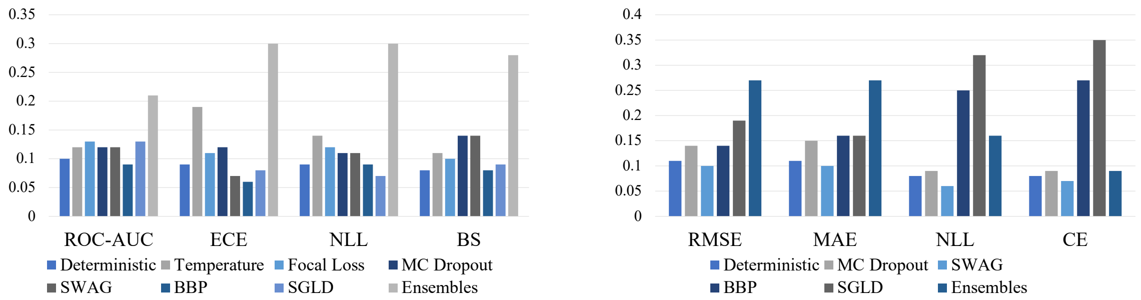
(a) Classification MRR ↑

(b) Regression MRR ↑

Figure 2.2: Mean Reciprocal Rank (MRR) of the backbone models.

2.4.2 Uncertainty Quantification

Figure 2.2 depict the uncertainty estimation performances via ECE, NLL, BS, and RCE columns. One discernible trend from Figure 2.2 is the consistent performance enhancement from Deep Ensembles even when the number of ensembles is limited, such as the QM9 case. MC Dropout also exhibits a similar trend, albeit less pronounced. Despite a possible compromise in prediction accuracy, TS emerges as another method that almost



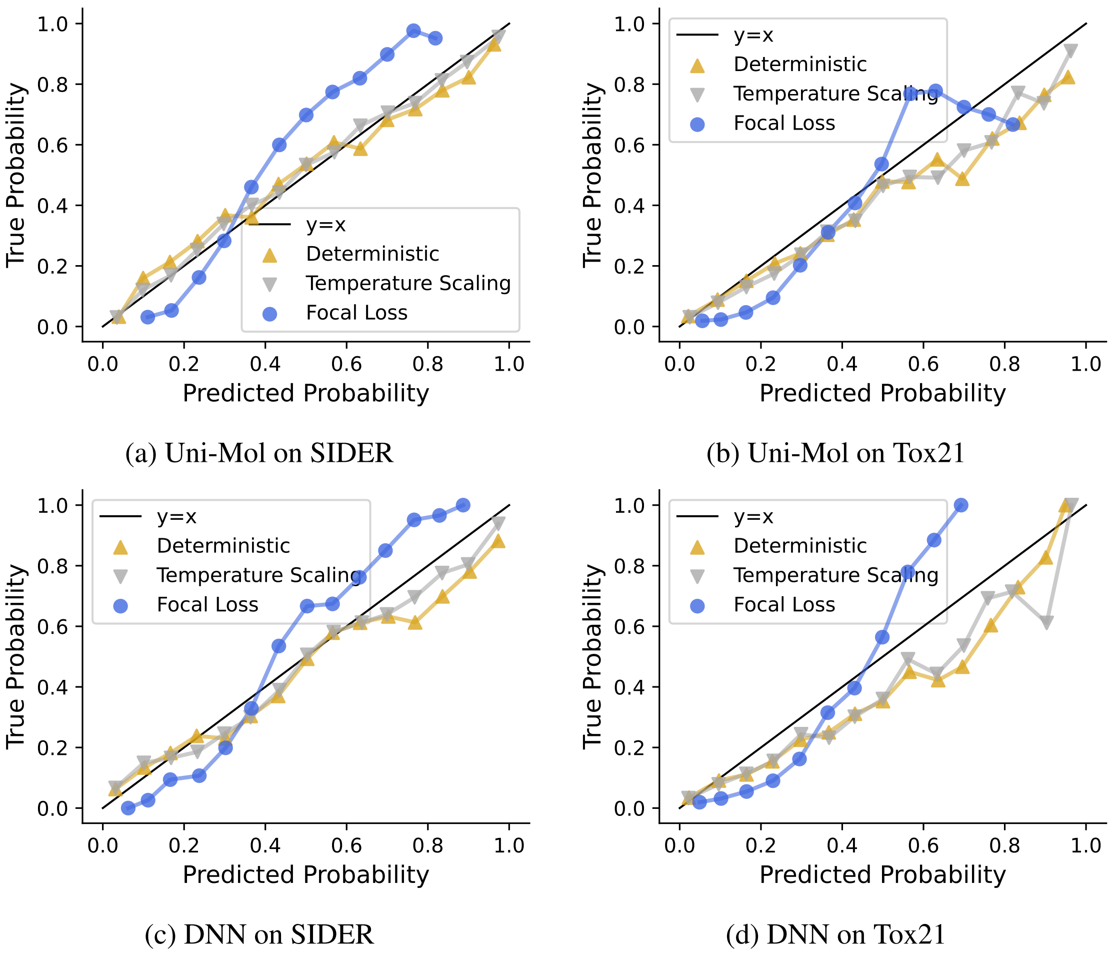
(a) Classification MRR ↑

(b) Regression MRR ↑

Figure 2.3: MRR of the UQ methods.

invariably improves calibration. Figure 2.4 shows that deterministic prediction tends to be over-confident, and TS mitigates this issue. An exception is noted in the ToxCast dataset, likely attributable to the distribution deviation between calibration and test datasets. However, it is susceptible to calibration-test alignment. If the correlation is weak, TS may worsen the calibration. In contrast, Focal Loss does not work as impressively for binary classification, which is due to parameters over-regularization that diminishes the prediction sharpness to an unreasonable value, a conjecture verified by the "S"-shaped calibration curves in Figure 2.4. Although limited in classification efficacy, both BBP and SGLD deliver commendable performance in predicting regression uncertainty. Yet, their inconsistent improvement of RMSE and MAE implies a greater influence on variance prediction than the mean. SGLD's tendency to "play safe" by predicting larger variances, while the deterministic method is prone to over-confident by ascribing small variances even to its inaccurate predictions. In addition, we do not observe a better correlation between SGLD's error and variance. We assume that the noisy training trajectory prevents SGLD and BBP from sufficiently minimizing the gap between the predicted mean and true labels, thus encouraging them to maintain larger variances to compensate for the error.

Figure 2.2 indicates that Uni-Mol exhibits subpar calibration, particularly for regression. Comparing Uni-Mol, ChemBERTa, and DNN in Figure 2.5, we notice that larger models such as Uni-Mol are more confident in their results, as illustrated by their smaller variances and larger portion of (std, error) points exceeding $y = kx$. Apart from the in-



(a) Uni-Mol on SIDER

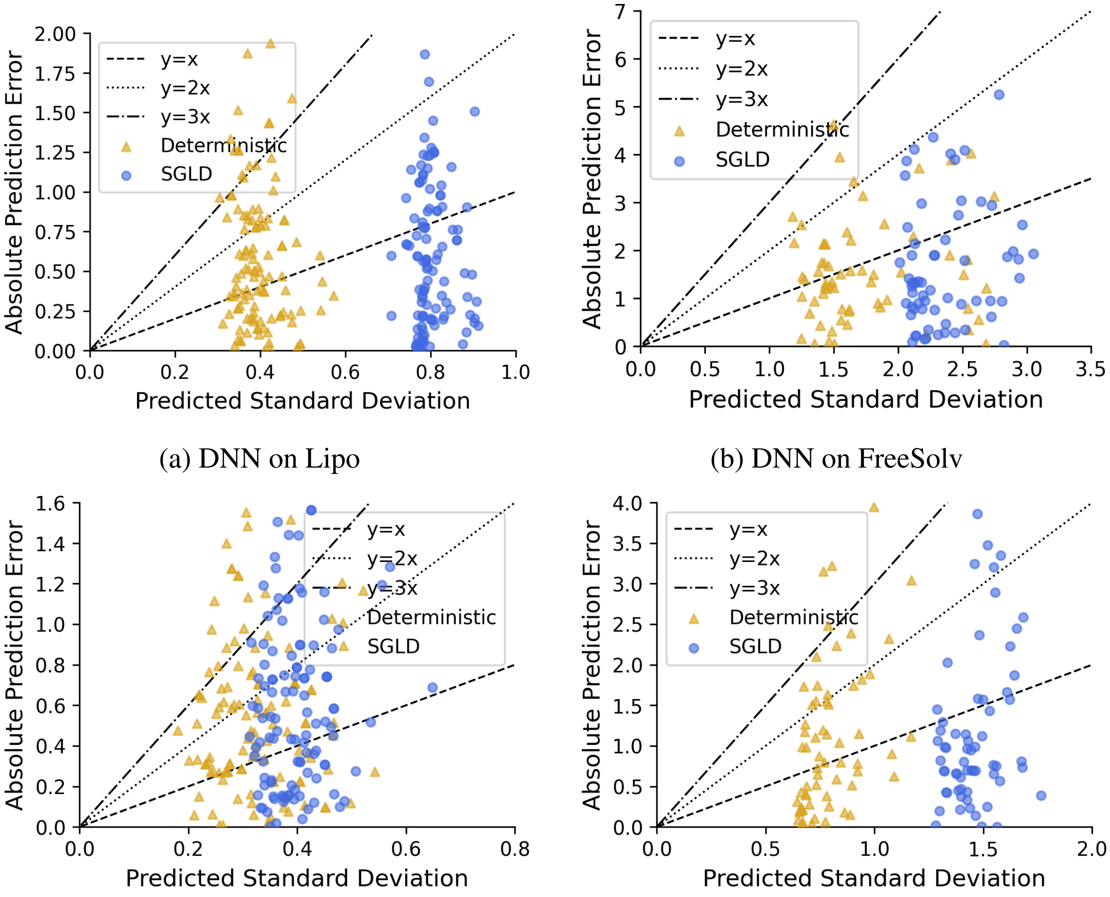
(b) Uni-Mol on Tox21

(c) DNN on SIDER

(d) DNN on Tox21

Figure 2.4: The calibration plot of Deterministic, Temperature Scaling and Focal Loss.

herent susceptibility of larger models to overfitting, such phenomenon for Uni-Mol could also be attributed to shared structural features in 3D conformations in the training and test molecules that remain unobserved in simpler descriptors [29]. While this similarity benefits property prediction, it also potentially misleads the model into considering data points as in-distribution, thereby erroneously assigning high confidence. Overall, our findings broadly support the notion that models with lower expressiveness tend to exhibit better calibration. Therefore, the selection of an appropriate UQ method is more critical for enhancing the calibration of larger models. This is particularly evident from the larger discrepancies in calibration errors between the Deterministic approach and the most effective UQ method for Uni-Mol compared to DNN.



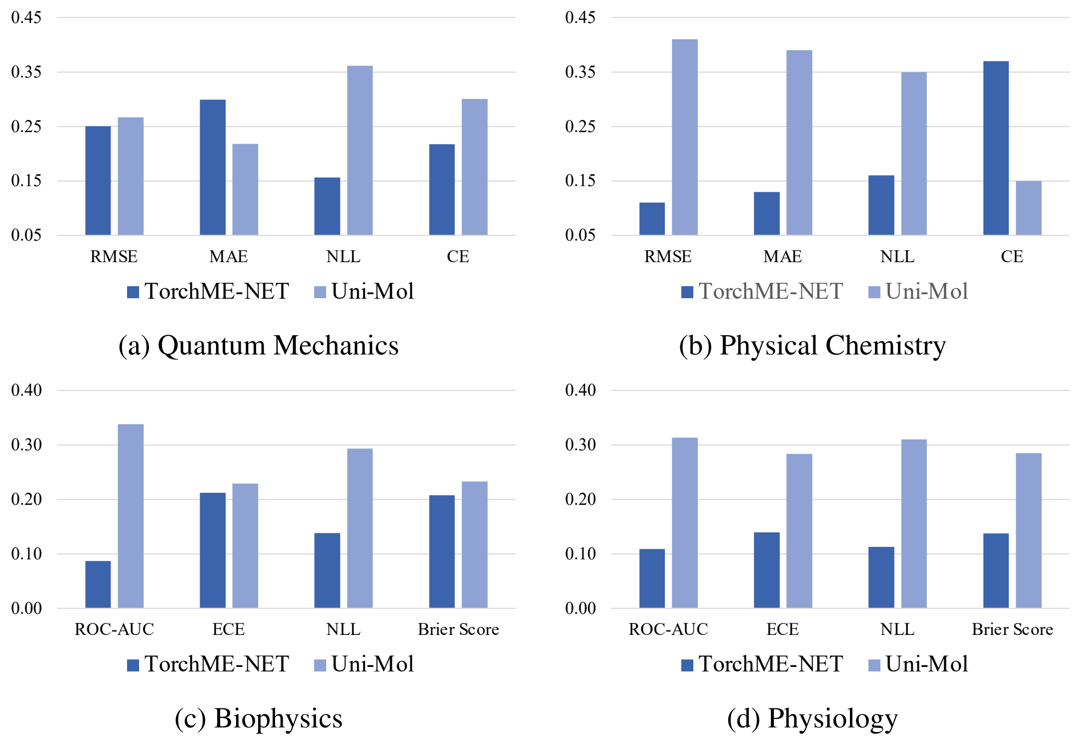
(a) DNN on Lipo

(b) DNN on FreeSolv

(c) ChemBERTa on Lipo

(d) Uni-Mol on FreeSolv

Figure 2.5: The absolute error between the model-predicted mean and true labels against the predicted standard deviation. We compare the performance of SGLD with the deterministic prediction on different backbones and datasets. The "$y = kx$" lines indicate whether the true labels lie within the k-std range of the predicted Gaussian. Also, a model is perfectly calibrated when its output points are arranged on an "$y = kx$" line for an arbitrary k. Notably, SGLD is observed to generate a larger variance for OOD samples, which tends to correspond more closely with the prediction errors on average.



(a) Quantum Mechanics

(b) Physical Chemistry

(c) Biophysics

(d) Physiology

Figure 2.6: MRRs of TorchMDNet and Uni-Mol on datasets grouped by dataset property categories. MRR calculations are confined to results from these two backbones. Only relative values matter. TorchMDNet is comparible to Uni-Mol on Quantum Mechanics properties, where it is pre-trained on, but is outperformed by Uni-Mol on all other property categories.

Table 2.6: Performance with frozen backbone weights and on random split datasets compared with the original scores in Table 2.4 and Table 2.5. The result is calculated as $(\text{new} - \text{original})/\text{original}$ and is macro-averaged over all datasets, backbones and UQ methods. The performance of the frozen backbone is significantly worse than the original, while the model performs better on the randomly split (in-domain) test set.

	Classification				**Regression**			
	ROC-AUC (%)↑	ECE (%)↓	NLL (%)↓	Brier Score (%)↓	RMSE (%)↓	MAE (%)↓	NLL (%)↓	CE (%)↓
Frozen Backbone	-24.07	145.13	53.43	88.98	78.60	92.40	58.18	-46.06
Random Split	13.25	-35.19	-33.87	-37.35	-26.34	-31.36	-53.87	19.06

2.4.3 TorchMD-NET and GIN

Our analysis also encompasses TorchMD-NET and GIN, two additional backbone models excluded from the primary benchmark due to their limited capabilities. As presented in the tables and Figure 2.6, TorchMD-NET's performance is on par with Uni-Mol when predicting quantum mechanical properties but falls short in others. This outcome aligns with expectations, given that TorchMD-NET's architecture is tailored specifically for predicting quantum mechanical properties [92]. Moreover, it is pre-trained on the relatively niche dataset PCQM4Mv2 with only the denoising objective, which might be suitable for molecular dynamics but limited for other properties. In contrast, Uni-Mol stands out as a versatile model, benefiting from diverse pre-training objectives that ensure superiority across various tasks. On the other hand, GIN's performance is consistently inferior to other models including DNN, with examples presented in Table 2.4 and Table 2.5, likely due to the limited expressiveness of 2D graphs and the GNN architecture when pre-training is absent.

2.4.4 Frozen Backbone and Randomly Split Datasets

Table 2.6 demonstrates a notable drop in prediction performance when backbone weights are fixed; and random splits outperform scaffold splits. This is consistent with intuition: if backbone models serve solely as feature extractors instead of a part of the trainable predictors, they are less expressive for downstream tasks. Additionally, in-distribution features tend to be more predictable. An interesting exception emerges in regression calibration error, where frozen backbones perform better and random splits score lower. Upon examining the predicted values, we note that predictions for random splits exhibit a sharper distribution, *i.e.*, smaller $\hat{\sigma}$. This suggests that the models are more confident in regressing in-distribution data, aligning with our previous observation for Uni-Mol. Conversely, frozen backbones are less prone to overfitting due to their constrained expressiveness. This behavior underscores the original models' capability to distinguish between in-distribution and OOD features and assign confidence scores with precision.

2.5 Conclusion

In [41] we present MUBen, a benchmark designed to evaluate a variety of UQ methods across different categories, using backbone models that employ various descriptors for multiple molecular property prediction tasks. The findings indicate that Deep Ensembles consistently enhance performance compared to the deterministic baseline, albeit at a substantial computational cost. For classification tasks, TS and MC Dropout are straightforward yet effective approaches. Conversely, for regression, BBP and SGLD appear more suitable in estimating uncertainty, although they may lead to a decrease in prediction accuracy, particularly with smaller backbone models. Among the different backbones, Uni-Mol stands out due to its effective utilization of 3D molecular conformations, which, while highly expressive, is also prone to overconfidence. Other backbone models, leveraging different descriptors, offer advantages under varying conditions and should be chosen based on the specific requirements of the use case.

Given the rapid advancements in molecular representation learning and UQ methods, MUBen cannot encompass all possible combinations, forcing us to focus on a curated selection of representative methods. Furthermore, we use coarse-grained hyperparameter grids to maintain experimental feasibility, which makes MUBen, while indicative of trends, might not present the best possible results. We remain committed to refining MUBen and welcome contributions from the broader community to enhance its inclusivity and utility for research in this field and related domains.

CHAPTER 3
LANGUAGE MODEL UNCERTAINTY QUANTIFICATION

3.1 Introduction

In the domain of NLP, autoregressive LLMs have increasingly permeated real-world applications, including high-stakes areas such as medical diagnosis [106, 107, 45], where errors can carry severe consequences. Although LLMs boast powerful generalization due to their vast and diverse pre-training data, they can falter when confronted with tasks or domains insufficiently represented in their training. Our work with Minesweeper [46] illustrates this point: once the model encounters significant distribution shifts, its reasoning degrades sharply, often defaulting to reiterations of seen examples or training data rather than genuinely adapting to new instructions [46]. Compounding this challenge is the phenomenon of hallucination [108, 109], wherein LLMs produce responses that appear convincing but contain factual inaccuracies. Left unchecked, such behaviors pose risks in settings where humans may unwittingly rely on the model's output for critical decisions.

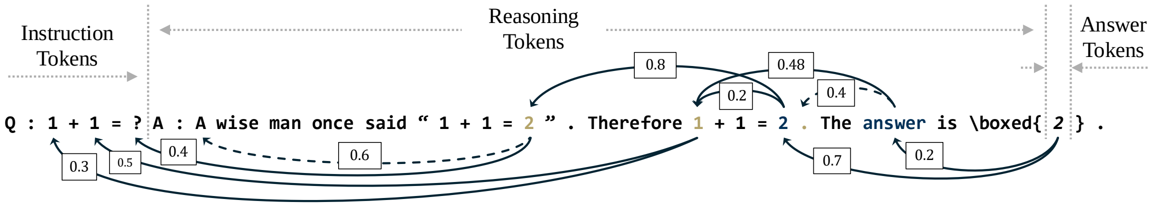
Consequently, reliable UQ becomes indispensable to gauge whether a model is overstepping its bounds of competence or generating spurious information. Unlike the simpler setting of discriminative autoencoding molecular PLMs discussed in the previous chapter, where established UQ techniques apply with minimal adjustments, LLMs generate open-ended token sequences in a vast language space, complicating the direct use of classical UQ methods. Tokens within a response are highly interdependent, and the autoregressive decoding process seldom aligns with the assumptions of traditional UQ frameworks [78, 110]. As a result, despite growing interest in confidence estimation for natural language generation, effective and efficient UQ methods tailored to autoregressive architectures remain limited, underscoring an urgent need for novel, robust solutions.

Earlier attempts at UQ for autoregressive models primarily involved cumulative token-wise probability or entropy calculations [111, 110], including length-normalized variants [112]. Nevertheless, these methods often conflated semantic and auxiliary tokens, leading to inefficiency and unreliability for long reasoning sequences. Addressing the intractability of the full reasoning space, several works approximated it through semantic representations of multiple model-generated outputs [110, 113, 114, 115, 116, 117]. Techniques such as semantic entropy [110], augmented-prompt ensembles [114], and input clarification [113] have shown promise but often incur high computational overhead due to multiple model inferences or reliance on external encoders.

Another direction explores isolating critical tokens in reasoning sequences, assessing their semantic contribution to the final output [118, 119]. Methods like SAR [118] and CSL [119] separate auxiliary from semantically significant tokens but either require multiple forward passes or large validation sets to select attention heads, complicating their practicality for extended outputs. Further research elicits explicit uncertainty statements from the model itself [120, 121, 122, 123], which is intuitive but often demands additional training or domain-specific fine-tuning, thus limiting broad applicability.

To address these gaps, we propose UQAC, a novel, model-agnostic framework that leverages *white-box* access to autoregressive language models. UQAC capitalizes on a simple yet powerful insight: not all tokens in a reasoning sequence contribute equally to the final answer. This aligns with prior findings on attention mechanisms in sequence modeling [124, 125, 13, 118, 119]. Concretely, UQAC constructs an *attention chain* by backtracking from the answer tokens through their attention weights to earlier tokens (Figure 3.1), isolating the most critical reasoning steps (subsection 3.3.1). Peripheral tokens are then pruned based on semantic similarity to the final answers (subsection 3.3.2) and probability thresholds (subsection 3.3.3), yielding a manageable set of key reasoning paths whose marginal probabilities can be computed directly.

Compared to existing UQ methods, UQAC stands out due to its:


Instruction Tokens
Reasoning Tokens
Answer Tokens
Q : 1 + 1 = ? A : A wise man once said “ 1 + 1 = 2 ” . Therefore 1 + 1 = 2 . The answer is \boxed{ 2 } .
0.8
0.48
0.2
0.4
0.4
0.6
0.3
0.5
0.7
0.2

	Step 1	Step 2	Step 3	Step 4
Source Tokens	{2}	{2, answer}	{2, 1}	∅
Target Tokens	{2, answer}	{2, 1, . *(excluded: below threshold)*}	{A *(excluded: stop word)*, ? *(instruction)*, 1 *(instruction)*, 1 *(instruction)*}	

Figure 3.1: Construction of the attention chain $\boldsymbol{x}_{\text{attn}}$ with a 3-step attention backtracking procedure. Arrows point from source tokens to target tokens with the top-2 attention weights (displayed on the arrows), *i.e.*, $l^{\text{tgt}} = 2$. Solid lines indicate valid target tokens, while dashed lines indicate invalid ones for various reasons explained below.

- **Applicability**: Compatible with any white-box autoregressive LLM, requiring neither additional training nor exhaustive hyperparameter searches.
- **Scalability**: Efficiently manages arbitrarily long outputs by focusing on only the most influential tokens.
- **Efficiency**: Integrates seamlessly with autoregressive generation without needing extra sampling or external encoders.
- **Calibration**: Produces bounded, interpretable confidence estimates ranging from 0 to 1.

Our experiments across multiple datasets and model architectures show that UQAC substantially improves calibration relative to comparable methods, all within a similar computational budget (section 3.5). Code and data for all experiments are available at https://github.com/Yinghao-Li/UQAC to facilitate reproducibility and further research.

3.2 Problem Definition

Consider an instruction sequence $\boldsymbol{x}_{\text{instr}} \in \mathbb{V}^{L_{\text{instr}}} \triangleq [x_1, x_2, \dots, x_{L_{\text{instr}}}]$ of length L_{instr}, where $\mathbb{V}$ is a vocabulary set. A language model $\mathcal{M}$ generates a response sequence $\boldsymbol{x}_{\text{resp}} \in \mathbb{V}^{L_{\text{resp}}}$ of length L_{resp}, which can be split into a Chain-of-Thought (CoT; [20]) reasoning sequence $\boldsymbol{x}_{\text{cot}} \in \mathbb{V}^{L_{\text{cot}}}$ and the final answer $\boldsymbol{x}_{\text{ans}} \in \mathbb{V}^{L_{\text{ans}}}$. We write $\boldsymbol{x}_{\text{resp}} = \boldsymbol{x}_{\text{cot}} \oplus \boldsymbol{x}_{\text{ans}}$ (Figure 3.1), where $\oplus$ represents "concatenation", disregarding any tokens after $\boldsymbol{x}_{\text{ans}}$.

Our objective in UQ is to determine the model's confidence in its final answer, denoted by $P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{instr}})$. Directly using the joint probability $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}}) = P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}_{\text{cot}}|\boldsymbol{x}_{\text{instr}})$ is not suitable, as the reasoning sequence $\boldsymbol{x}_{\text{cot}}$ typically includes both semantically crucial tokens needed for deriving the answer and auxiliary tokens that primarily serve syntactic or coherence purposes. These auxiliary tokens often make up the bulk of $\boldsymbol{x}_{\text{cot}}$ yet have minimal impact on the actual answer semantics. Consequently, using $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}})$ as a confidence measure can be both biased and unintuitive: the probability tends to decrease monotonically with the length of the response, leading to small joint probabilities that are difficult to interpret. Moreover, conditioning the final answer's confidence on $\boldsymbol{x}_{\text{cot}}$ is unnecessary when the primary concern is the correctness and trustworthiness of the final answer itself. Therefore, this confidence formally is expressed as

$$P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{instr}}) \;=\; \sum_{\mathbf{x}_{\text{cot}}} P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\mathbf{x}_{\text{cot}}, \boldsymbol{x}_{\text{instr}})\, P_{\mathcal{M}}(\mathbf{x}_{\text{cot}}|\boldsymbol{x}_{\text{instr}}), \tag{3.1}$$

where $\mathbf{x}_{\text{cot}}$ ranges over all possible reasoning sequences in $\mathbb{V}^{L_{\text{cot}}}$. However, the summation is computationally intractable: even when the vocabulary dimension is on the order of 10,000, the space grows exponentially with the sequence length L_{cot}, which can reach into the hundreds. Accordingly, the main challenge in LLM UQ is to construct an accurate approximation $\widetilde{P}_{\mathcal{M}} \;\approx\; P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{instr}})$ that faithfully reflects the model's confidence, without explicitly summing over all possible reasoning sequences.

In the following, we focus on a single instance of a single model and thus omit the model indicator $\mathcal{M}$ and the instance index. Table B.1 summarizes the notations used throughout.

3.3 Uncertainty Quantification with Attention Chain

3.3.1 Attention Chain

We first apply *attention backtracking* to identify semantically crucial tokens $\boldsymbol{x}_{\text{attn}} \sqsubset \boldsymbol{x}_{\text{cot}}$, , where $\sqsubset$ is defined as a "proper (not necessarily consecutive) subsequence", *i.e.*, $\boldsymbol{x}_1 \in \mathbb{V}^N \sqsubset \boldsymbol{x}_2 \in \mathbb{V}^M \leftrightarrow \exists 1 \leqslant i_1 < \cdots < i_N \leqslant M$ *s.t.* $x_{1,[k]} = x_{2,[i_k]}, \forall k \in \mathbb{N}_{[1,N]}$. Attention backtracking itself is an iterative procedure governed by a function $f\colon \boldsymbol{x}_{\text{src}} \mapsto \boldsymbol{x}_{\text{tgt}}$, which selects a target subsequence $\boldsymbol{x}_{\text{tgt}} \sqsubset \boldsymbol{x}_{\text{cot}}$ deemed most influential for predicting the source tokens $\boldsymbol{x}_{\text{src}} \sqsubset \boldsymbol{x}_{\text{resp}}$, based on the model's attention weights. We impose a buffer size $L_{\text{tgt}} \triangleq \max|\boldsymbol{x}_{\text{tgt}}|$ that bounds the number of tokens selected at each step. The process starts with $\boldsymbol{x}_{\text{src}}^{(0)} = \boldsymbol{x}_{\text{ans}}$ and $\boldsymbol{x}_{\text{attn}}^{(0)} = \emptyset$. At each iteration z, we compute $\boldsymbol{x}_{\text{tgt}}^{(z)} = f(\boldsymbol{x}_{\text{src}}^{(z-1)})$ and update $\boldsymbol{x}_{\text{src}}^{(z)} = \boldsymbol{x}_{\text{tgt}}^{(z)}$. This continues until no new tokens are extracted from $\boldsymbol{x}_{\text{cot}}$. The final sequence $\boldsymbol{x}_{\text{attn}}$ is then the concatenation of all target sequences $\boldsymbol{x}_{\text{attn}} \triangleq \boldsymbol{x}_{\text{tgt}}^{(0)} \oplus \cdots \oplus \boldsymbol{x}_{\text{tgt}}^{(Z)}$, assuming f is applied Z times. $\boldsymbol{x}_{\text{attn}}$ serves as a compact semantic approximation of the original reasoning chain $\boldsymbol{x}_{\text{cot}}$. Figure 3.1 provides an illustration of attention backtracking. Each step of f consists of three stages: 1) attention weight processing; 2) attention head selection and aggregation; and 3) target token identification, detailed in the following paragraphs.

Attention Weight Processing For a sequence of length T, LLMs employ self-attention [13] to compute a weight for each token in the prefix $\boldsymbol{x}_{\leqslant T}$ (which includes the instruction, *i.e.*, $\boldsymbol{x}_{\text{instr}} \sqsubset \boldsymbol{x}_{\leqslant T}$) to gauge its contribution to predicting the next token x_{T+1}. In an L-layer, H-head Transformer model $\mathcal{M}$, the attention weight vector for token x_T in the l-th layer and h-th head is defined as

$$\boldsymbol{\alpha}_T^{(l,h)} \in [0,1]^T = \text{softmax}(\frac{\boldsymbol{K}_T^{(l,h)} \boldsymbol{q}_T^{(l,h)}}{\sqrt{d_k}}), \tag{3.2}$$

where $\boldsymbol{q}_T^{(l,h)} \in \mathbb{R}^{d_k}$ is the query vector for x_T, $\boldsymbol{K}_T^{(l,h)} \in \mathbb{R}^{T \times d_k}$ is the key matrix for the prefix $\boldsymbol{x}_{\leqslant T}$, and d_k is the dimensionality of the key vectors. This attention vector determines how much semantic information from $\boldsymbol{x}_{\leqslant T}$ is transferred to x_{T+1}, thereby serving as an indicator of the relative importance of each token in the prefix.

However, LLMs often overemphasize tokens that are close to position T, irrespective of their semantic relevance. To mitigate this bias, we reweight the most recent C attention weights by multiplying them with a sequence of decreasing factors $\boldsymbol{\gamma} \in [0,1]^C$. Moreover, since some LLMs tend to assign disproportionately high attention to the instruction's BOS token, we reset that particular attention weight to zero. After these modifications, the attention vector is renormalized as follows (with layer and head indices omitted for brevity):

$$\boldsymbol{\alpha}_{T,[T-C:T]} \leftarrow \boldsymbol{\gamma} \odot \boldsymbol{\alpha}_{T,[T-C:T]}; \quad \alpha_{T,[1]} \leftarrow 0; \qquad \boldsymbol{\alpha}' = \frac{\boldsymbol{\alpha}}{\sum \boldsymbol{\alpha}}, \tag{3.3}$$

where "$\odot$" denotes element-wise multiplication and the square brackets indicate a slice. Please refer to subsection B.2.1 for further discussion on the reweighting factors $\boldsymbol{\gamma}$.

Attention Head Selection and Aggregation Different attention heads typically capture distinct aspects of the input sequence. Not all attention heads contribute equally when predicting x_{T+1}, making it crucial to identify the most informative ones for discerning semantically significant tokens [119]. To enable a *lightweight*, *training-free* selection process, we leverage attention entropy, defined as $\mathcal{H}(\boldsymbol{\alpha}) = -\sum \boldsymbol{\alpha} \log \boldsymbol{\alpha}$, which quantifies the uncertainty of the attention distribution. A higher entropy value indicates a more uniform distribution, suggesting a less informative attention head. Consequently, we select the top-K heads with the lowest entropy:

$$(l,h)_{T,k} = \underset{(l,h)}{\arg\min}_k \, \mathcal{H}(\boldsymbol{\alpha}_T'^{(l,h)}); \quad k \in \mathbb{N}_{[1,K]}, \tag{3.4}$$

where the operator $\arg\min_k$ returns the *index* corresponding to the k-th smallest entropy value, and the subscript T emphasizes that the computation is performed independently for each token x_T. Subsequently, we aggregate the selected attention weights into a single vector $\boldsymbol{\alpha}^*_T \in [0,1]^T$ via element-wise maximization:

$$\alpha^*_{T,[t]} = \max_k \alpha'^{(l,h)_{T,k}}_{T,[t]}; \quad t \in \mathbb{N}_{[1,T]}. \tag{3.5}$$

Due to the maximization, $\sum \boldsymbol{\alpha}^*_T \in (1, T]$, *i.e.*, $\boldsymbol{\alpha}^*_T$ no longer resides on a probability simplex.

Target Token Identification Let $t_s \in \mathbb{N}_{[1,T]}$ for $s \in \mathbb{N}_{[1,L_{\text{src}}]}$ denote the positions of source tokens within the source token set $\boldsymbol{x}_{\text{src}} \triangleq \{x_{t_s}\}_s$ (with step index $\cdot^{(z)}$ omitted). Here we treat $\boldsymbol{x}_{\text{src}}$ and $\boldsymbol{x}_{\text{tgt}}$ as *sets* for convenience. Given the set of attention weights $\{\boldsymbol{\alpha}^*_{t_s}\}_s$ corresponding to $\boldsymbol{x}_{\text{src}}$, our goal is to construct $\boldsymbol{x}_{\text{tgt}}$ by selecting the top-L_{tgt} tokens based on their cumulative attention weights. For each token position $t \in \mathbb{N}_{[1,T]}$, we compute a cumulative attention weight and gather the target set as

$$\begin{gathered}\phi_{[t]} = \sum_{s=1}^{L_{\text{src}}} \alpha^*_{t_s,[t]}; \\ \boldsymbol{x}_{\text{tgt}} = \{x_t \mid \phi_{[t]} > \theta;\ \mathbb{I}(x_t \notin \mathbb{V}_{\text{stop}});\ t \in \{\arg\max_t{}_k\, \phi_t\}_{k=1}^{L_{\text{tgt}}}\}.\end{gathered} \tag{3.6}$$

where θ is a threshold to filter out weakly attended tokens, and the operator $\arg\max_k$ returns the indices corresponding to the k-th largest cumulative weight.

Finally, we define the step-wise backtracking function f as the sequential composition of Equation 3.2–Equation 3.6. The attention chain $\boldsymbol{x}_{\text{attn}}$ is generated by iteratively applying f; at each step z, a new target set is produced, and the final attention chain is the concatenation of all these target sets, as discussed above.

When transferring the target set $\boldsymbol{x}_{\text{tgt}}$ to serve as the new source $\boldsymbol{x}_{\text{src}}$ for the next iteration (*i.e.*, updating $\boldsymbol{x}^{(z)}_{\text{src}} = \boldsymbol{x}^{(z)}_{\text{tgt}}$ for iteration $z+1$), it is important to differentiate between *token generation* and *token attention*. During token generation, a new token x_{T+1} is produced based on the input token x_T, with the attention computation operating between x_T and the preceding context $x_{\leqslant T}$. In contrast, during token attention, the projected key and value vectors are computed directly from the token x_t as it is fed into the model.

Suppose that x_{T+1} is identified as a semantically crucial token and we wish to trace back the tokens that contributed to its prediction via attention backtracking. In this case, we focus on the attention vectors generated during the token's *generation*.

$$\boldsymbol{\alpha}_T \in [0,1]^T = \operatorname{softmax}\left(\frac{\boldsymbol{K}_T \boldsymbol{q}_T}{\sqrt{d_k}}\right). \tag{3.7}$$

Assume further that a token $x_t \in \boldsymbol{x}_{\text{tgt}}$ is the most attended token in the target set, *i.e.*,

$$\alpha_{T,[t]} = \max_i \alpha_{T,[i]} = \max_i \boldsymbol{k}_{T,i}^{\mathsf{T}} \boldsymbol{q}_T, \tag{3.8}$$

where $\boldsymbol{k}_{T,i} \triangleq \boldsymbol{K}_{T,[i]}$ denotes the i-th key vector in the key matrix $\boldsymbol{K}_T$ corresponding to token x_i. In this scenario, x_t serves as a model input (generated from x_{t-1}) rather than an output.

Consequently, when propagating the target set to the next iteration, the indices are shifted back by one position $\boldsymbol{x}_{\text{src}}^{(z)} = \{x_{i-1} \mid x_i \in \boldsymbol{x}_{\text{tgt}}^{(z)}\}$. The same index-shifting principle applies to the initial source set $\boldsymbol{x}_{\text{src}}^{(0)} = \{x_{i-1} \mid x_i \in \boldsymbol{x}_{\text{ans}}\}$.

3.3.2 Similarity-Based Filtering

Since the backtracking function f does not explicitly regulate the attention chain length L_{attn}, the chain may be long, complicating the formation of a tractable reasoning space. Therefore, we employ a similarity-based filtering strategy for finer control. Given the answer tokens $\boldsymbol{x}_{\text{ans}}$ and the attention chain $\boldsymbol{x}_{\text{attn}}$, we compute the similarity between each token pair (x_m, x_n), where $x_m \sqsubset \boldsymbol{x}_{\text{ans}}$ and $x_n \sqsubset \boldsymbol{x}_{\text{attn}}$, by applying cosine similarity to their last-layer output logits $\boldsymbol{h}_m$ and $\boldsymbol{h}_n$. We calculate the similarity vector $\boldsymbol{w} \in [-1, 1]^{L_{\text{attn}}}$ as:

$$\begin{aligned} \text{sim}(x_m, x_n) &= \frac{\boldsymbol{h}_m^\top \boldsymbol{h}_n}{\|\boldsymbol{h}_m\| \|\boldsymbol{h}_n\|}, \\ w_{[n]} &= \sum_m \text{sim}(x_m, x_n), \quad m \in \mathbb{N}_{[1,L_{\text{ans}}]},\ n \in \mathbb{N}_{[1,L_{\text{attn}}]}. \end{aligned} \tag{3.9}$$

We retain the tokens with the top-L'_{attn} weights to form the filtered sequence $\boldsymbol{x}'_{\text{attn}} \sqsubseteq \boldsymbol{x}_{\text{attn}}$:

$$\boldsymbol{x}'_{\text{attn}} = [x_{\text{attn},[n_1]}, \dots, x_{\text{attn},[n_i]}, \dots]\ \textit{s.t.}\ w_{[n_i]} > 0;\ n_i \in \{\arg\max_k \boldsymbol{w}\}_{k=1}^{L'_{\text{attn}}}, \tag{3.10}$$

where the threshold of 0 excludes tokens with negative similarity scores.

Our similarity calculation is practical and efficient as it leverages model $\mathcal{M}$'s hidden states $\boldsymbol{h}$ rather than external Sentence-BERT embeddings [126, 110, 116]. Because these hidden states are produced “for free” during autoregressive decoding, no additional computational overhead is incurred. In contrast, extracting Sentence-BERT embeddings would require multiple extra forward passes through a separate model that employs a different vocabulary and tokenization scheme. Moreover, due to the autoregressive nature of $\mathcal{M}$, the hidden states corresponding to the same token at different positions are not necessarily identical:

$$\boldsymbol{h}_m \neq \boldsymbol{h}_n \quad \text{and} \quad 0 < \boldsymbol{h}_m^\top \boldsymbol{h}_n < 1, \quad \forall m, n\ \textit{s.t.}\ x_m = x_n, m \neq n, \tag{3.11}$$

This property helps reduce the risk of inadvertently filtering out short indicator tokens, such as “yes”, “no”, or multiple-choice options, when relying on external embeddings.

3.3.3 Model Confidence

Using the attention chain $\boldsymbol{x}_{\text{attn}}$ and its filtered version $\boldsymbol{x}'_{\text{attn}}$, we first define two confidence approximations in Equation 3.1 based on the joint probabilities of the attention chain and answer tokens: **1) attention approximation** $\widetilde{P}_{\mathcal{M},\text{attn}} \triangleq P(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}_{\text{attn}} | \boldsymbol{x}_{\text{instr}})$; **2) similarity**

approximation $\widetilde{P}_{\mathcal{M},\text{sim}} \triangleq P(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}'_{\text{attn}}|\boldsymbol{x}_{\text{instr}})$. Since $\boldsymbol{x}'_{\text{attn}} \sqsubseteq \boldsymbol{x}_{\text{attn}}$, we have $\widetilde{P}_{\mathcal{M},\text{sim}} \geq \widetilde{P}_{\mathcal{M},\text{attn}}$. Both approximations can be viewed as indicators of the model's confidence in the critical semantics captured by the final answer $\boldsymbol{x}_{\text{ans}}$. By marginalizing out the filtered attention chain $\boldsymbol{x}'_{\text{attn}}$ in $\widetilde{P}_{\mathcal{M},\text{sim}}$, we obtain **3) answer approximation** $\widetilde{P}_{\mathcal{M}} \triangleq P(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{instr}})$, which more directly captures the model's confidence in $\boldsymbol{x}_{\text{ans}}$ alone. The challenge is to reduce the approximated reasoning space $\mathbb{V}^{L'_{\text{attn}}}$ to a manageable subset $\mathbb{S}$ so that the summarization in

$$\widetilde{P}_{\mathcal{M}} = \sum_{\mathrm{x}'_{\text{attn}} \sim \mathbb{S}} P(\boldsymbol{x}_{\text{ans}}, \mathrm{x}'_{\text{attn}}|\boldsymbol{x}_{\text{instr}}) \tag{3.12}$$

remains computationally feasible. Approximating $\boldsymbol{x}_{\text{cot}}$ by $\boldsymbol{x}'_{\text{attn}}$ significantly reduces the token count, and we constrain $L'_{\text{attn}} \leq 10$. Furthermore, language models typically assign very high probabilities (*e.g.*, > 0.99) to the selected tokens; hence, substituting a chosen token $\boldsymbol{x}'_{\text{attn},[i]}$ with any other candidate $\mathrm{x}'_{\text{attn},[i]} \neq x'_{\text{attn},[i]}$ often results in a negligible joint probability, *i.e.*, $P\big(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}'_{\text{attn},[\backslash i]}, \mathrm{x}'_{\text{attn},[i]} \neq x'_{\text{attn},[i]}|\boldsymbol{x}_{\text{instr}}\big) \approx 0$. Consequently, we only consider alternative tokens that exceed a 0.01 probability threshold, and we alter just one token at a time while keeping the others unchanged. In our experiments, this yields on average only six such candidates per position (excluding $\boldsymbol{x}'_{\text{attn}}$), so $\mathbb{E}[|\mathbb{S}|] = 7$. Although multiple forward passes are still required, all elements in $\mathbb{S}$ are independent and can be evaluated without any recurrent or sequential dependencies, in contrast to Self-Consistency approaches.

3.4 Experiment Setup

3.4.1 Datasets and Models

We evaluate our approach using three widely recognized reasoning datasets that span various domains, including mathematical problem solving, logical reasoning, and commonsense reasoning. Only *test* partitions of the datasets are used for evaluation.

- **GSM8K** [50]: A dataset comprising 8,792 (7,473 for training and 1,319 for test) high-quality grade school math word problems that require multi-step reasoning.

Each problem includes a question and a detailed, step-by-step solution.

- **MATH** [51]: A collection of 12,500 (7,500 for training and 5,000 for test) challenging competition-level math problems covering subjects such as algebra, geometry, calculus, and more. Each problem is paired with a detailed solution.
- **BIG-Bench Hard (BBH)** [127, 128]: A subset of BIG-Bench consisting of 23 tasks identified as particularly challenging for LLMs, summarizing to 6,511 test instances in total. These tasks span domains such as logical reasoning, mathematics, and commonsense reasoning. In addition to providing the correct answers, BBH includes detailed CoT reasoning annotations for each question, thereby enabling the evaluation of both final answers and intermediate reasoning processes.

We evaluate 9 instruction-tuned white-box SOTA LLMs spanning 1B–9B parameters: Llama-3.2-1B and 3B, and Llama-3.1-8B [129]; gemma-2-2B and 9B [130]; Qwen2.5-1.5B, 3B, and 7B [131], and DeepSeek-R1-Distill-Llama-8B [25]. These models generally cover the most widely adopted instruction-tuned LLMs on the smaller side in practical use cases.

For the mathematical reasoning tasks on GSM8K and MATH, we employ a *zero-shot* prompting strategy without additional formatting instructions, ensuring that models generate responses directly from the input questions. In most cases, the answers from all models are straightforwardly extractable. However, gemma-2 models require additional guidance; they are explicitly instructed to enclose their final answers within a `\boxed{}` wrapper to facilitate easier extraction. For BBH, we utilize 3 provided in-context examples, supplemented with explicit instructions and modified prompts. These modifications direct the models to encapsulate their final answers within the `\boxed{}` wrapper, ensuring consistent and reliable answer extraction across the dataset. During our experiments, we set the maximum sequence length to 1,024 tokens for GSM8k and MATH, and to 1,536 tokens for BBH to accommodate the in-context examples in the latter dataset. For DeepSeek-R1, we further extend this limit by an additional 512 tokens to incorporate the “deep thinking” tokens. Please check section B.3 for additional details about dataset processing.

3.4.2 Baselines

We compare the three variants of our proposed method, UQAC, namely $\widetilde{P}_{\mathcal{M},\text{attn}}$, $\widetilde{P}_{\mathcal{M},\text{sim}}$, and $\widetilde{P}_{\mathcal{M}}$, against a diverse set of baselines. These baselines are selected to cover a broad spectrum of uncertainty quantification techniques, incorporating both token-level and response-level approaches. In particular, we consider the following methods:

1. $P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{cot}}, \boldsymbol{x}_{\text{instr}})$: The joint conditional probability of the answer, conditioned on both the CoT reasoning and sequence and the instruction sequence.
2. $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}})$: The joint conditional probability of the entire response given the instruction.
3. $\overline{P}_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}})$: The mean conditional probability computed over the answer tokens.
4. $\overline{P}_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}})$: The mean conditional probability computed over all tokens in the response.
5. **Predictive Entropy** $\mathcal{H}$ [110]: A token-level uncertainty measure that aggregates entropy over the response, where the value do not have a fixed range:

$$\mathcal{H} \in [0, L_{\text{resp}}] = -\sum_{t=L_{\text{instr}}+1}^{L_{\text{instr}}+L_{\text{resp}}} \sum_{\mathrm{x}_t \sim \mathbb{V}} P_{\mathcal{M}}(\mathrm{x}_t|\boldsymbol{x}_{<t}) \log P_{\mathcal{M}}(\mathrm{x}_t|\boldsymbol{x}_{<t}). \tag{3.13}$$

6. **Length-Normalized Predictive Entropy** $\overline{\mathcal{H}}$ [112]: It normalizes the predictive entropy by the response length to account for variations in output size:

$$\overline{\mathcal{H}} \in [0, 1] = \frac{\mathcal{H}}{L_{\text{resp}}}. \tag{3.14}$$

7. **Self-Consistency** [132]: The probability averaged over 5 independently sampled answers, thereby reflecting the consistency of the model's outputs.
8. **Verbalized Uncertainty** [122]: A model-generated confidence score obtained via additional prompting.

The first six baselines derive their uncertainty estimates directly from token probabilities, while the latter two rely on aggregating information from multiple or additional model outputs to capture uncertainty at a higher level. It is important to note that earlier approaches, such as the semantic entropy methods proposed in [110, 116], provide only a weaker approximation of Self-Consistency. These methods focus on capturing lexical diversity but do so at the expense of substantially higher computational overhead. In addition, the semantic ambiguity are minimal in our reasoning tasks, making these methods less relevant. Consequently, we neglect such methods.

3.4.3 Evaluation Metrics and Implementation Details

Prior works produce unbounded confidence scores [110, 118] and thus rely on AUROC for its scale-invariance. While AUROC is effective in evaluating the discriminative ability of a classifier, *i.e.* its capacity to separate correct from incorrect predictions, it does not assess the calibration of the predicted confidence scores. Calibration refers to the degree of agreement between the predicted probabilities and the actual likelihoods of the outcomes. For instance, consider a scenario with three instances having predicted confidence scores of 0.9, 0.5, and 0.1, where the first two predictions are correct and the third is incorrect. In this case, AUROC would yield a score of 1, indicating perfect separation. However, the same AUROC score of 1 would result even if the predictions were assigned extremely low and uninformative confidence values (*e.g.*, 9×10^{-3}, 8×10^{-10}, and 7.99×10^{-10}), despite the fact that such scores lack meaningful interpretation. Consequently, a high AUROC does not guarantee that the confidence estimates reflect the true probability of correctness. If a model outputs a maximum estimated confidence of only 1×10^{-3} across 10,000 instances, users might be misled about the model's overall certainty despite an ostensibly perfect AUROC.

Consequently, when queries originate from distributions different from those seen during training or from previously tested scenarios, relying solely on AUROC yields limited

Table 3.1: Performance of different UQ methods. The scores (in $\%$) are reported as $\mu \pm \sigma$ over 5 runs, averaged across all LLMs. Higher AUROC is better; ECE the opposite.

	GSM8k		**MATH**		**BBH**	
	AUROC↑	ECE↓	AUROC↑	ECE↓	AUROC↑	ECE↓
$\overline{P}_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}})$	60.9±0.6	49.4±0.0	74.2±1.2	48.1±0.2	65.0±0.9	44.6±0.3
$\overline{P}_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}})$	61.1±0.8	41.9±0.0	69.0±1.5	41.5±0.1	59.3±1.3	42.6±0.2
$P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}} \vert \boldsymbol{x}_{\text{cot}}, \boldsymbol{x}_{\text{instr}})$	60.7±0.6	48.4±0.1	73.9±1.2	43.7±0.3	66.1±0.8	38.1±0.5
$P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}} \vert \boldsymbol{x}_{\text{instr}})$	66.7±0.7	50.0±0.0	79.0±1.1	50.0±0.0	63.6±1.5	45.3±0.4
$1 - \overline{\mathcal{H}}$[(a)]	61.8±0.8	33.2±0.1	69.3±1.5	35.3±0.2	59.4±1.3	33.5±0.4
$1 - \mathcal{H}$[(a)]	67.1±0.8	-	78.6±1.1	-	63.6±1.2	-
Self-Consistency	66.4±1.9	28.9±0.8	79.5±1.0	15.8±0.8	79.5±1.0	31.6±0.7
Verbalized	54.9±0.5	42.9±0.2	57.4±0.7	45.1±0.2	58.2±1.2	39.7±0.3
UQAC-$\widetilde{P}_{\mathcal{M},\text{attn}}$	58.4±0.8	37.4±0.6	68.6±1.2	42.8±0.5	64.6±1.3	23.4±1.0
UQAC-$\widetilde{P}_{\mathcal{M},\text{sim}}$	60.7±1.0	28.0±0.5	68.0±1.3	21.6±1.0	65.1±1.2	22.1±1.0
UQAC-$\widetilde{P}_{\mathcal{M}}$	61.3±0.9	33.6±0.4	69.5±1.2	25.8±0.9	66.7±1.2	24.2±0.9

[(a)] We use $1-\mathcal{H}$ and $1-\overline{\mathcal{H}}$ as higher entropy indicates higher uncertainty, different from probabilities. ECE is not applicable to predictive entropy $\mathcal{H}$ as its value is unbounded.

insight into the true reliability of the model's answers. Hence, we emphasize calibration metrics, specifically ECE [90] and calibration plots with 20 bins. For completeness, we also report AUROC in our results. Details regarding the calculation of AUROC and ECE are introduced previously in subsection 2.3.4 and section 2.2.

We set $C = 10$ in Equation 3.3, $K = 16$ in Equation 3.4, $L_{\text{tgt}} = 3$ and $\theta = 0.5$ in Equation 3.6, and delay applying θ until $L_{\text{attn}} \geqslant 5$ to avoid premature backtracking. We use $L'_{\text{attn}} = 10$ in Equation 3.10. For metrics, we balance correct and incorrect predictions by subsampling. All experiments run on an NVIDIA A100-SXM4-80GB GPU with bf16 precision. Results, reported as mean (μ) ± standard deviation (σ) over five runs (seeds 0–4), are from our own implementations.

3.5 Results

3.5.1 Main Results

Table 3.1 summarizes the average performance across all LLMs, showing superior calibration from UQAC. Notably, the unnormalized baselines, *i.e.*, $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}})$ and $\mathcal{H}$, achieve higher AUROC on GSM8k and MATH by leveraging pronounced differences in response lengths between correct and incorrect answers (shown in Figure 3.3a and discussed in subsection 3.5.7). Since longer sequences yield lower joint probabilities and higher cumulative entropy, this effect benefits the AUROC calculation. However, when response lengths are more balanced, as with DeepSeek-R1 on GSM8k (Figure 3.3c) or across most models on the BBH dataset due to in-context regularization, the advantage diminishes. Furthermore, $\mathcal{H}$ cannot be calibrated, and $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}})$ suffers from a 50.0% ECE, indicating that predicted probabilities are pushed toward extremes (1 as in Figure 3.2a or 0 as in Figure 3.2b). This concentration harms the ability to distinguish correct from incorrect predictions based solely on confidence, yielding poor calibration. While the length-normalized predictive entropy $\overline{\mathcal{H}}$ performs comparably to UQAC on GSM8k, it falters with longer inputs and extended reasoning chains, as evidenced by its inferior performance on the MATH and BBH datasets and a more centralized entropy distribution (Figure 3.2c). In general, UQAC provides descent AUROC scores as well as delivers significantly lower ECE across datasets, aligning more closely with true predictive performance and offering more reliable uncertainty estimates (Figure 3.2).

3.5.2 External Baselines

Without additional fine-tuning, Verbalized Uncertainty exhibits significantly inferior performance, consistently underperforming even the $\overline{\mathcal{H}}$ baseline. This suggests that it is not effectively integrated into prevalent LLMs and requires extensive training or careful prompt engineering for practical utility. In contrast, Self-Consistency, a variant of Deep Ensem-

Table 3.2: Performance differences when applying similarity-based filtering Equation 3.10 to all response tokens $\boldsymbol{x}_{\text{resp}}$ instead of only the attention chain L_{attn}, *i.e.*, $\boldsymbol{x}_{\text{attn}} = \boldsymbol{x}_{\text{resp}}$.

		Llama-3.2-1B		gemma2-2b		Qwen2.5-1.5B		Average	
		AUC ↑	ECE ↓	AUC ↑	ECE ↓	AUC ↑	ECE ↓	AUC ↑	ECE ↓
GSM8k	UQAC	64.75	19.54	68.10	35.45	57.97	34.76	63.61	29.92
	w/o attns	61.56	22.06	66.17	32.22	60.09	35.22	62.61	29.83
MATH	UQAC	67.92	20.65	74.09	15.23	71.71	31.45	71.24	22.45
	w/o attn	65.25	24.55	70.30	18.94	72.65	31.57	69.40	25.02
BBH	UQAC	61.91	21.05	60.78	26.01	65.90	21.35	62.86	22.80
	w/o attn	62.17	22.52	60.06	25.60	62.25	22.82	61.49	23.65

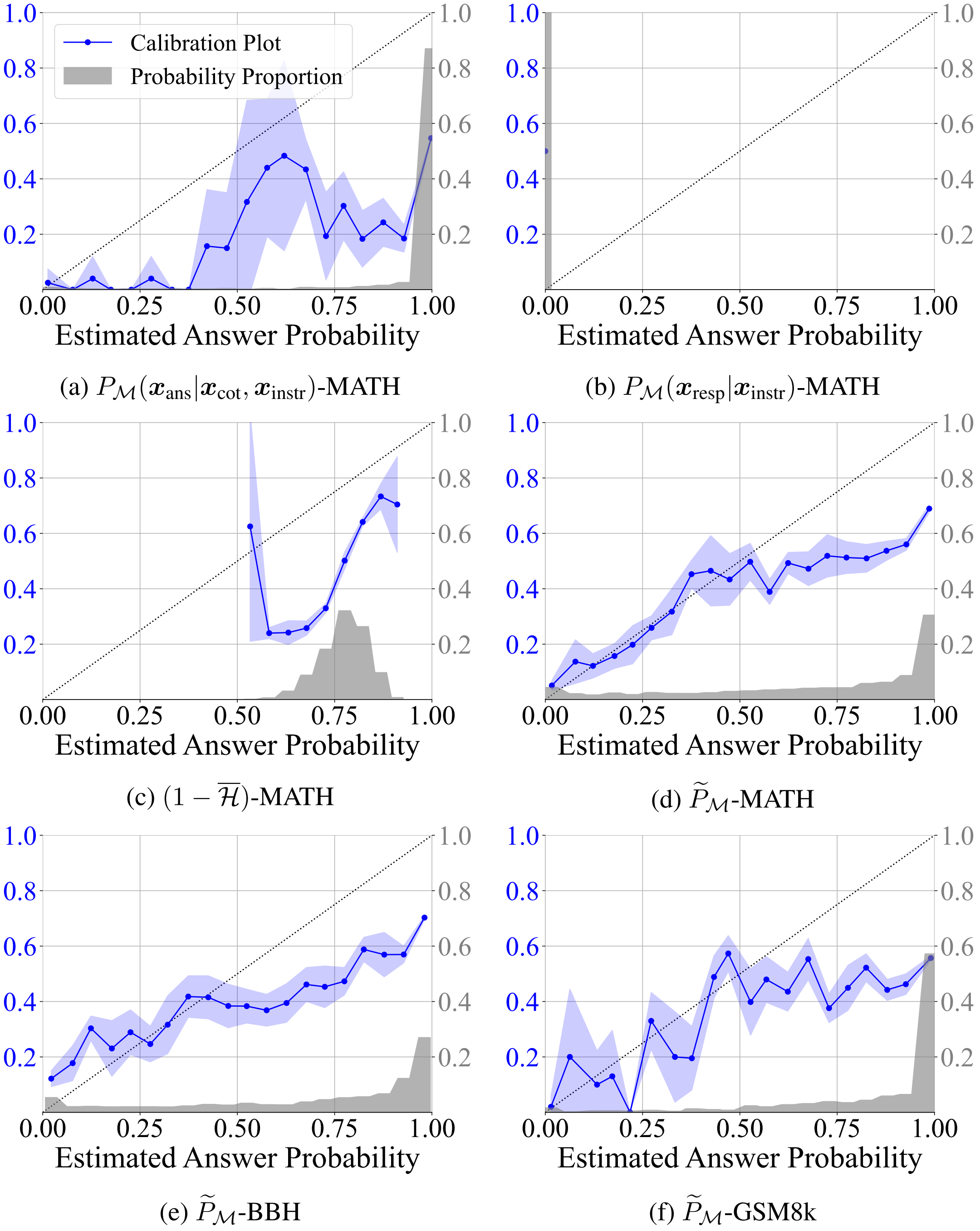
bles [80], generally achieves superior performance, albeit with significant computational overhead (shown in Figure 3.6 and discussed in subsection 3.5.8), aligning with theoretical insights and prior empirical evidence [104, 132, 41]. When efficiency is not a primary concern, it is possible to combine Self-Consistency with UQAC to further enhances calibration performance.

3.5.3 UQAC Variants

Among the variants of UQAC, $\widetilde{P}_{\mathcal{M},\text{attn}}$ yields lower AUROC and higher ECE. As reasoning becomes more complex and the attention chain L_{attn} lengthens, the predicted probabilities concentrate around zero, leading to increased ECE. This highlights the need to control token count via similarity filtering to preserve a balanced confidence distribution. Conversely, $\widetilde{P}_{\mathcal{M}}$, developed from $\widetilde{P}_{\mathcal{M},\text{sim}}$, consistently outperforms $\widetilde{P}_{\mathcal{M},\text{sim}}$ in terms of AUROC, albeit at the expense of higher ECE. Thus, the optimal variant depends on the specific application requirements.

3.5.4 Ablation Study

Since similarity filtering Equation 3.10 effectively controls the size of the estimated attention space $\mathbb{S}$, and given that L_{attn} inherently has higher similarity to $\boldsymbol{x}_{\text{ans}}$ compared to $\boldsymbol{x}_{\text{resp}}$ (Figure 3.5, subsection 3.5.7), we investigate whether using only similarity scores without



(a) $P_{\mathcal{M}}(\boldsymbol{x}_{\text{ans}}|\boldsymbol{x}_{\text{cot}}, \boldsymbol{x}_{\text{instr}})$-MATH

(b) $P_{\mathcal{M}}(\boldsymbol{x}_{\text{resp}}|\boldsymbol{x}_{\text{instr}})$-MATH

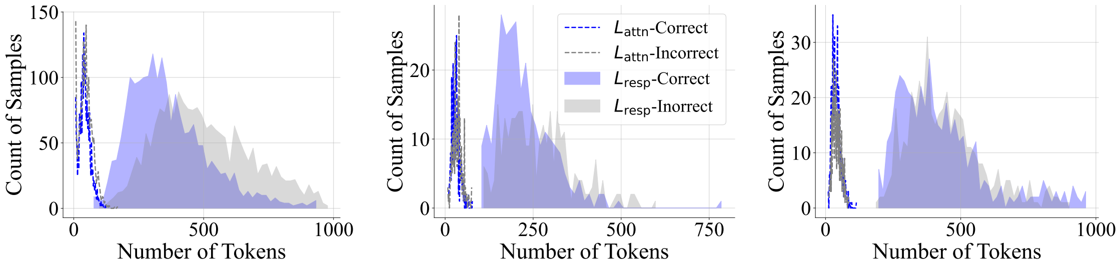
(c) $(1 - \overline{\mathcal{H}})$-MATH

(d) $\widetilde{P}_{\mathcal{M}}$-MATH

(e) $\widetilde{P}_{\mathcal{M}}$-BBH

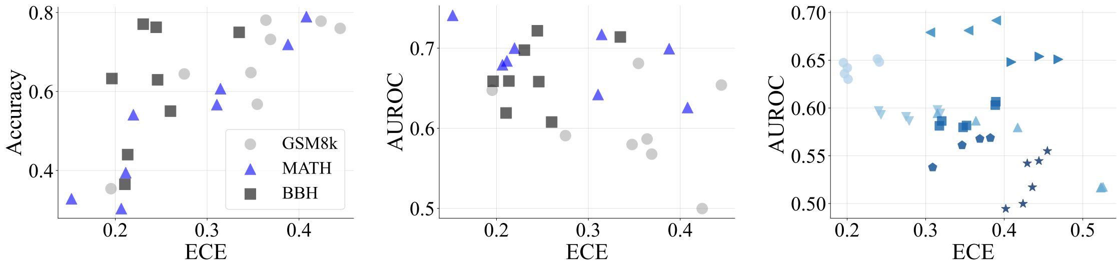
(f) $\widetilde{P}_{\mathcal{M}}$-GSM8k

Figure 3.2: Calibration plots and probability histogram for Llama-3.1-8B. The x-axis shows the centers of 20 probability bins. The calibration curve (blue line with $\mu \pm \sigma$) displays actual accuracy per bin, while the gray shadow represents the probability proportion.



(a) Llama-3.1-8B-MATH (b) Llama-3.1-8B-GSM8k (c) DeepSeek-R1-8B-GSM8k

Figure 3.3: Histograms of response and attention chain lengths.



(a) ECE-Accuracy model (b) ECE-AUC model (c) ECE-AUC hyper-parameter

Figure 3.4: Correlation analysis of ECE with accuracy and AUROC for UQAC-$\widetilde{P}_{\mathcal{M}}$. In Figure 3.4a and Figure 3.4b, different point types differentiate dataset; in Figure 3.4c they distinguish models.

the attention backtracking described in subsection 3.3.1 is sufficient. Specifically, we set $\boldsymbol{x}_{\text{attn}} = \boldsymbol{x}_{\text{resp}}$ in subsection 3.3.2, ignoring subsection 3.3.1. Table 3.2 compares the generalizable results of this similarity-only variant against UQAC on smaller-scale models. The similarity-only variant performs notably worse than UQAC, exhibiting higher ECE and lower AUROC scores. This indicates that attention backtracking is crucial for effectively identifying semantically important tokens and enhancing calibration performance.

3.5.5 Correlation Analysis

Previous studies such as [41] report a negative correlation between a model's prediction accuracy and its UQ performance, suggesting that higher accuracy often coincides with poorer calibration. To verify whether this holds for autoregressive LLMs, Figure 3.4a plots each model's ECE against its accuracy across all datasets. The strong correlation shown in the figure, evidenced by an average Pearson's coefficient of 0.760, confirms that the conclusion still holds for LLMs. This is further discussed in subsection 3.5.6 from the dataset perspective.

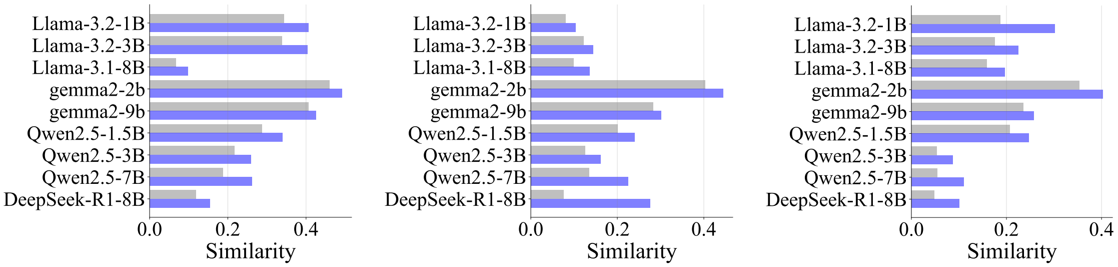
We further analyze the relationship between ECE and AUROC under two conditions. Across different models (presented by the same type of points), Figure 3.4b reveals a negligible correlation (average coefficient of –0.147), indicating that calibration and AUROC capture distinct performance aspects and should be evaluated jointly, echoing section 3.4. However, when examining $\widetilde{P}_{\mathcal{M}}$ with hyper-parameters, specifically, varying L'_{attn} from 8 to 12 and the similarity threshold in Equation 3.10 between 0 and 0.2, a more pronounced positive correlation (coefficient of 0.489) emerges. These findings suggest a trade-off between calibration and AUROC, warranting further exploration in future work.

3.5.6 Potential Overfitting

On GSM8k, nearly all uncertainty quantification methods show a significant performance gap compared to those on more challenging datasets. As illustrated in Figure 3.2f versus Figure 3.2d and Figure 3.2e, GSM8k exhibits a marked overconfidence: the probability distribution is highly centralized around 1 and the calibration curve deviates substantially from the ideal diagonal. This behavior suggests that SOTA LLMs may be overfitting on GSM8k, thereby questioning its reliability as a benchmark for evaluating LLM capabilities.

3.5.7 Response Lengths

Figure 3.3 illustrates the lengths of the entire response sequences L_{resp} and the attention chain L_{attn}, categorized by answer correctness. The lengths of the similarity-filtered chain (L'_{attn}) are omitted as they are controlled. Across all models, L_{attn} is less than 10% of L_{resp}, demonstrating that the attention backtracking function f effectively reduces computational overhead. Additionally, while L_{resp} is noticeably longer for incorrect predictions, this difference is less pronounced for L_{attn}. Given that tokens in the attention chain have a higher



(a) GSM8k

(b) MATH

(c) BBH

Figure 3.5: UQ method efficiency and average token similarity between the answer tokens $\boldsymbol{x}_{\text{ans}}$ and 1) the attention chain tokens $\boldsymbol{x}_{\text{attn}}$ (blue bars) and 2) the response tokens $\boldsymbol{x}_{\text{resp}}$ (gray bars). The similarity scores are computed by averaging the cosine similarities of token embeddings over all examples. It shows that $\boldsymbol{x}_{\text{attn}}$ tokens are inherently more similar to $\boldsymbol{x}_{\text{ans}}$ tokens than average $\boldsymbol{x}_{\text{resp}}$ tokens, indicating that the attention chain is more likely to capture the answer-relevant context.

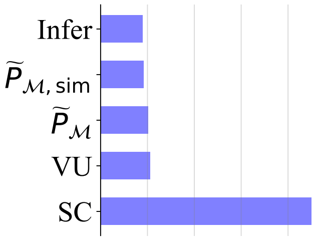


Figure 3.6: Average computational overhead (running time, excluding I/O) measured on the Llama-3.1-8B model when generating outputs of approximately 500 tokens. “Infer” refers to the baseline inference time needed to generate responses; “VU” denotes the Verbalized Uncertainty approach; and “SC” indicates Self-Consistency with 5 sampled responses.

average similarity to the answer tokens (shown in Figure 3.5 with $\text{sim}(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}_{\text{attn}}) = 0.257$ versus $\text{sim}(\boldsymbol{x}_{\text{ans}}, \boldsymbol{x}_{\text{resp}}) = 0.206$), these results confirm that f successfully captures the reasoning process and identifies critical tokens.

3.5.8 Computational Overhead

Compared to standard inference in Figure 3.6, UQAC introduces only a slight computational overhead, demonstrating higher efficiency compared to the Verbalized Uncertainty method. In contrast, although Self-Consistency achieves superior calibration performance, it substantially increases computational requirements. This overhead makes Self-Consistency less feasible for real-time applications, particularly in scenarios demanding extensive output generation or strict latency constraints.

3.6 Conclusion

This work addresses UQ for LLMs, particularly in tasks where final answers are derived from the intermediate reasoning steps. We propose UQAC, a method that leverages attention chains constructed according to the autoregressive attention weights to efficiently identify and track semantically critical tokens within the reasoning sequence, significantly reducing the space for uncertainty estimation. UQAC provides reliable uncertainty estimates with minimal computational overhead, requiring only a few parallel forward inferences without additional fine-tuning, multiple recurrent response sampling, or depending on external models. Empirical evaluations across diverse benchmarks and model architectures confirm that UQAC achieves superior calibration, especially when intermediate reasoning steps substantially influence final outcomes. Moreover, UQAC is applicable to any white-box autoregressive LLMs, preserves reasoning interpretability, and scales efficiently with increasing model size and complexity.

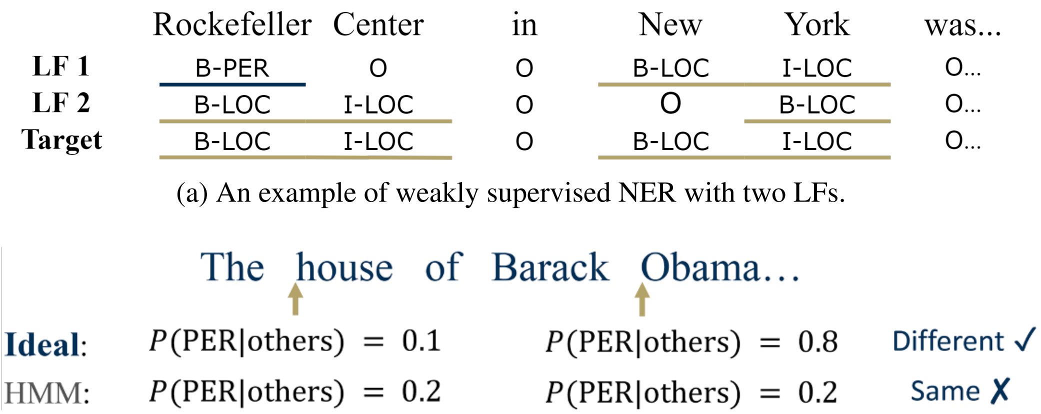
CHAPTER 4
DATA-EFFICIENT INFORMATION EXTRACTION SYSTEMS

4.1 Introduction

In the previous chapters, we have studied and developed various UQ methods for both autoencoding and autoregressive PLMs, spanning tasks in general as well as scientific domains. Such UQ methods are central for ensuring the safety and reliability of PLMs when deployed in real-world applications. Building upon these uncertainty quantification techniques, the subsequent chapters illustrate how to utilize model/function reliability estimates to enhance performance across diverse settings. They also discuss how to design prompting strategies or fine-tuning procedures to maximize performance gains from limited datasets.

4.1.1 Information Extraction and Named Entity Recognition

IE is a fundamental NLP task aimed at extracting structured information from unstructured text. It finds application in domains such as biomedical research [133], clinical data analysis [134], materials property extraction [52], social media analysis [135], and customer feedback processing [136], *etc.* A pivotal sub-task is NER, wherein pre-defined named entities are identified in text. NER has wide-reaching implications, underpinning tasks such as knowledge graph construction, question answering, and material synthesis [137]. While fully supervised NER has historically delivered strong performance, it relies extensively on large, high-quality labeled datasets [49]. This dependency makes it expensive and time-consuming to create new corpora or update existing ones with additional entity types or expanded document collections [57, 59, 48].


Rockefeller Center in New York was...
LF 1 B-PER O O B-LOC I-LOC O...
LF 2 B-LOC I-LOC O O B-LOC O...
Target B-LOC I-LOC O B-LOC I-LOC O...
The house of Barack Obama...
Ideal: P(PER|others) = 0.1 P(PER|others) = 0.8 Different ✓
HMM: P(PER|others) = 0.2 P(PER|others) = 0.2 Same ✗

(a) An example of weakly supervised NER with two LFs.

(b) Illustration of the disadvantage of HMMs.

Figure 4.1: An example of weakly supervised NER with two LFs, along with the issue of appling HMMs to weakly supervised NER.

4.1.2 Weak Supervision

To mitigate the high cost of manual annotations, *weak supervision* has gained attention as a promising alternative [47, 56]. Instead of meticulously labeling each token, multiple, easier-to-obtain LFs generate weak, often noisy labels [55, 56]. These LFs may utilize domain-specific dictionaries, pattern-matching rules, or knowledge bases [57], automatically labeling large volumes of data. Since they do not depend on manually annotated gold data, LFs can facilitate zero-shot NER when properly designed.

However, deploying LFs introduces new complexities. Different LFs often exhibit varying levels of coverage and precision. They can also conflict with each other, occasionally providing contradictory labels for the same token (see Figure 4.1a). Approaches like Majority Voting (MV) and Snorkel [56] simply assign labels token by token, overlooking global context. To address this, more advanced frameworks [138, 139, 57, 140] represent true labels as latent variables in a graphical model and perform unsupervised learning to reconcile inconsistent LF outputs. In particular, HMMs have been explored to capture sequential dependencies among tokens [138, 139, 57], though the Markov property constrains them to adjacent dependencies and tends to overlook broader sentence semantics. Subsequent endeavors often train a deep neural network on the aggregated weak labels [141, 49],

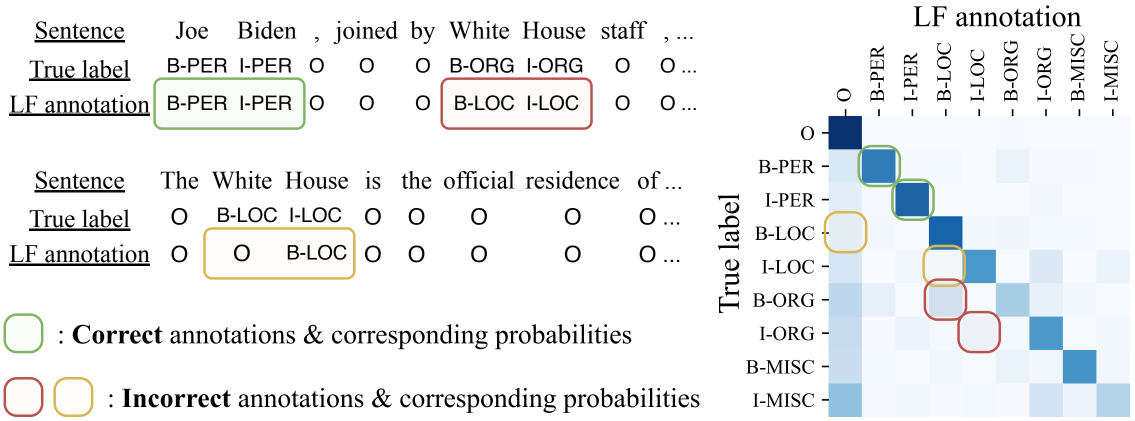

Sentence Joe Biden , joined by White House staff , ...
True label B-PER I-PER O O O B-ORG I-ORG O O ...
LF annotation B-PER I-PER O O O B-LOC I-LOC O O ...
Sentence The White House is the official residence of ...
True label O B-LOC I-LOC O O O O O ...
LF annotation O O B-LOC O O O O O ...
: Correct annotations & corresponding probabilities
: Incorrect annotations & corresponding probabilities
LF annotation
True label
O
B-PER
I-PER
B-LOC
I-LOC
B-ORG
I-ORG
B-MISC
I-MISC

Figure 4.2: An illustration of weak annotations for weakly supervised NER and the corresponding true emission pattern in an HMM.

hoping to refine these noisy labels with powerful end models. Nevertheless, [49] reveals that the end model does not always outperform the label aggregator, indicating that label aggregation quality remains a central challenge.

4.1.3 Conditional Hidden Markov Model

In [59], we propose CHMM to overcome the limited expressiveness of classical HMMs in the weakly supervised NER setting. Unlike conventional HMMs with fixed transition and emission matrices, CHMM predicts token-wise transition and emission probabilities by conditioning the observation marginals on domain-specific autoencoding PLM embeddings. The token-wise probabilities allows CHMM to adjust the transitions and emissions according to the input tokens, which is more flexible than HMM's constant counterpart (see Figure 4.1b for an illustration). This design introduces greater flexibility for modeling how true labels evolve with diverse token contexts, thereby alleviating the strict Markov constraint.

Yet, CHMM faces several obstacles when scaling up: 1) it allocates a full emission matrix for each LF and label combination, leading to a huge parameter space and a non-convex optimization landscape; 2) it lacks a clear mechanism to disentangle LF reliability from off-diagonal errors, which complicates interpreting and managing LF mistakes.

4.1.4 Sparse Conditional Hidden Markov Model

To address these scalability and interpretability concerns, [60] introduces Sparse-CHMM. It retains CHMM's conditioning on BERT for transition probabilities but adopts a more compact parameterization for emission probabilities (see Figure 4.2). The core insight is that *the diagonal each element of an emission matrix reflect LF reliability in its prediction, while off-diagonal elements capture the likelihood of incorrect labels*. The diagonal emission elements bear the most significance as they directly parameterize the LF performance. Hence, Sparse-CHMM focuses first on learning each LF's reliability score before expanding that score into a complete emission matrix through carefully designed expansion functions. Compared to CHMM, the fewer parameters in Sparse-CHMM lead to more stable training and better scalability.

Nonetheless, certain LFs may systematically produce errors or label tokens in conflicting ways. To handle such cases, Sparse-CHMM incorporates the Weighted XOR (WXOR) scores, which capture mutual disagreement frequencies among LFs and refine the off-diagonal emission probabilities when needed. Additionally, Sparse-CHMM leverages a Dirichlet-based sampling scheme to balance deterministic reliability priors with stochastic noise, mitigating local optima. Extensive experiments confirm that Sparse-CHMM advances SOTA performance in weakly supervised NER while maintaining interpretability and efficiency.

4.1.5 Autoregressive Information Extraction

Although weak supervision considerably reduces labeling overhead, its effectiveness depends on the domain relevance of heuristics and LFs, as well as tuning hyperparameters for different tasks. Moreover, both fully and weakly supervised methods often assume a fixed set of entity types and relations, limiting their applicability in scenarios where new or diverse information categories may emerge.

With the rise of LLMs, attention has shifted toward IE methods that require minimal

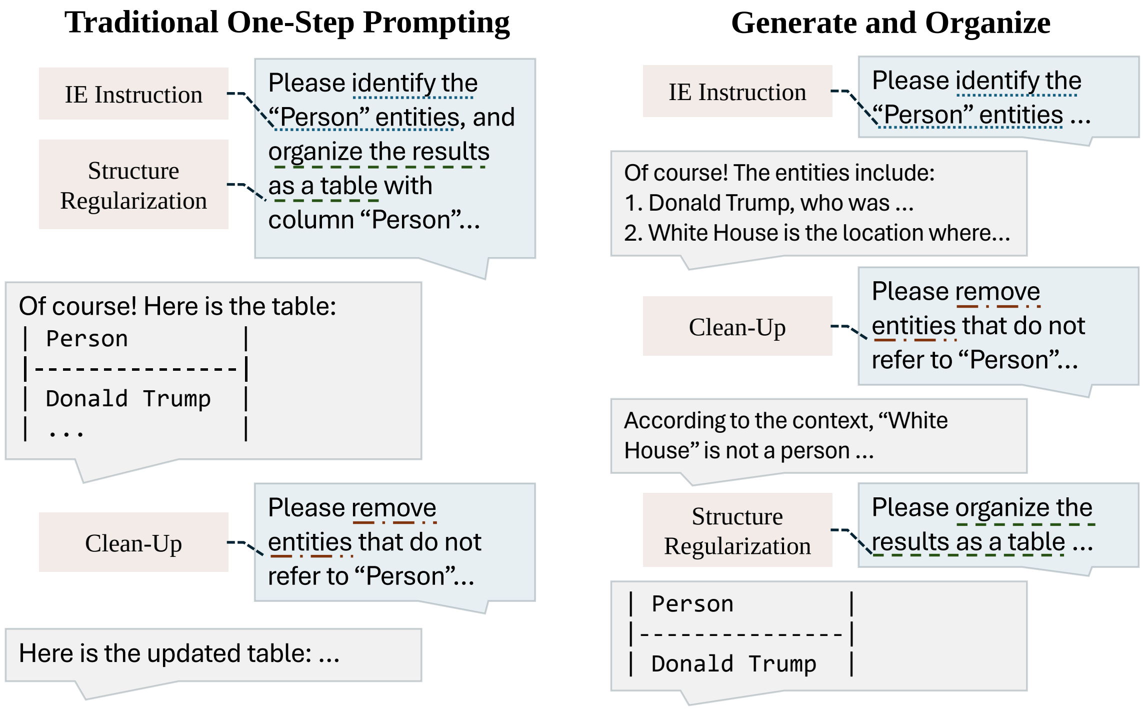

Traditional One-Step Prompting
IE Instruction
Structure Regularization
Please identify the “Person” entities, and organize the results as a table with column “Person”...
Of course! Here is the table: | Person | |---------------| | Donald Trump | | ... |
Clean-Up
Please remove entities that do not refer to “Person”...
Here is the updated table: ...
Generate and Organize
IE Instruction
Please identify the “Person” entities ...
Of course! The entities include: 1. Donald Trump, who was ... 2. White House is the location where...
Clean-Up
Please remove entities that do not refer to “Person”...
According to the context, “White House” is not a person ...
Structure Regularization
Please organize the results as a table ...
| Person | |---------------| | Donald Trump |

Figure 4.3: The pipeline of G&O-NER, compared with One-Step prompting methods.

task-specific labels or domain knowledge. Promising strategies include prompting LLMs directly [142, 143, 144, 145] or fine-tuning them using either gold labels or pseudo labels generated by GPTs [146, 147, 148, 149, 150]. However, a key challenge is guiding the model to produce well-structured outputs from unstructured, natural language prompts. Dedicated prompting strategies and output templates [147, 150] can help, though they risk format inconsistencies or performance degradation when multiple instructions must be combined.

In this thesis, we propose a straightforward yet effective approach, G&O, for enabling LLMs to perform structured zero-shot IE tasks, focusing on both NER and Relation Extraction (RE). Our method splits the output generation process into two primary steps: 1) a free-form generation stage where the model naturally explains and extracts relevant information; and 2) a structuring stage that subsequently converts this free-form text into a standardized output format. We further introduce a clean-up module to remove extraneous noise in the free-form responses prior to the structuring step, mitigating potential misalign-

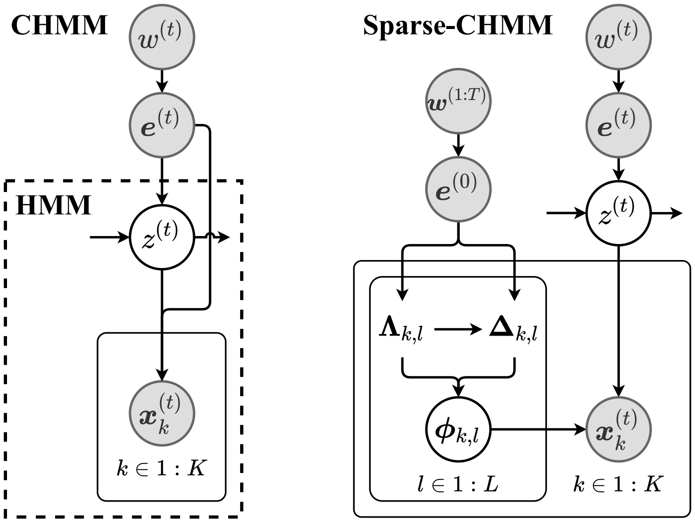


Figure 4.4: Architectures of HMM variants. $w^{(t)}$ is the input token; $\boldsymbol{e}^{(t)}$ is the corresponding embedding; $\boldsymbol{x}_k^{(t)}$ is the observation from LF k; $z^{(t)}$ is the hidden state; $\boldsymbol{\Lambda}$ and $\boldsymbol{\Delta}$ are base and addon prior matrices; and $\boldsymbol{\Phi}$ is the sampled emission matrix.

ment between instructions and the desired structured output. Empirical results demonstrate that G&O consistently improves zero-shot IE performance across different LLMs. Each component—free-form generation, clean-up, and structuring—contributes to these gains. Additionally, G&O can be easily combined with self-consistency techniques [132] or other post-processing approaches to meet more complex formatting requirements.

To foster research on both weak supervision and generative IE, we provide code for CHMM at https://github.com/Yinghao-Li/CHMM-ALT, Sparse-CHMM at https://github.com/Yinghao-Li/Sparse-CHMM, and G&O at https://github.com/Yinghao-Li/GnO-IE. We hope these resources prove valuable for researchers and practitioners, enabling further innovations in efficient and flexible IE methods.

4.2 Problem Setup

4.2.1 Weakly Supervised Named Entity Recognition

Suppose we have T categorical tokens $\boldsymbol{w}^{(1:T)}$ in the input sentence and a set of candidate entities $\mathcal{E}$ with size E, discriminative NER taggers assign one entity label to each token

$w^{(t)}, t \in \mathbb{N}_{[1,T]}$. Using the BIO tagging scheme, the label set is $\mathbb{L} = \{\texttt{O}\} \cup \{\texttt{B}\text{-ent}, \texttt{I}\text{-ent}\}_{\text{ent}\in\mathcal{E}}$ with the size of $L = 2E + 1$, where "O" (out-of-definition) indicates unrelated entities.

For weakly supervised NER, we have K independent LFs, each providing a sequence of weak annotations $\boldsymbol{x}_k^{(1:T)}$. $\boldsymbol{x}_k^{(t)} \in \{0, 1\}^L$ is one-hot over the label set $\mathbb{L}$. Label models aim to approximate the ground-truth labels $\boldsymbol{y}^{(1:T)} \in \mathbb{L}^T$ with latent states $\boldsymbol{z}^{(1:T)} \in \mathbb{L}^T$ given the input tokens and weak annotations $\{\boldsymbol{w}^{(1:T)}, \boldsymbol{x}_{1:K}^{(1:T)}\}$. We do not distinguish the label string and label index, and uniformly represent them by integers l, i, or $j \in \mathbb{N}_{[1,L]}$. In addition, we focus on one sentence and omit the sentence index $m \in \mathbb{N}[1, M]$ unless specified otherwise.

4.2.2 Autoregressive Information Extraction

In autoregressive IE, we adopt a flexible approach that does not rely on a predefined set of tokens and labels. Rather, we encapsulate the context, represented as $\boldsymbol{w}_{\text{instr}}$, within a prompt that articulates the task using natural language, as illustrated in Figure 4.3. The ultimate output of the LLM is a response sequence $\boldsymbol{w}_{\text{resp}}$, typically adhering to a predetermined format like a Markdown table, JSON, or a list, which is dictated by the initial prompt. To achieve this objective, we may require one or more round of LLM interactions. This response structure is intentionally chosen to facilitate post-processing, enabling the efficient extraction of targeted information such as entities and relationships from the model's output.

4.3 Hidden Markov Models

CHMM and Sparse-CHMM model true entity labels as hidden variables, inferring them from observed noisy labels. As depicted in Figure 4.4 and Figure 4.5, traditional discrete HMMs rely on a single transition and emission matrix to model the probabilities of all label transitions and hidden state-observation emissions. These matrices remain static once trained. Contrastingly, CHMM conditions both its transition and emission matrices on the

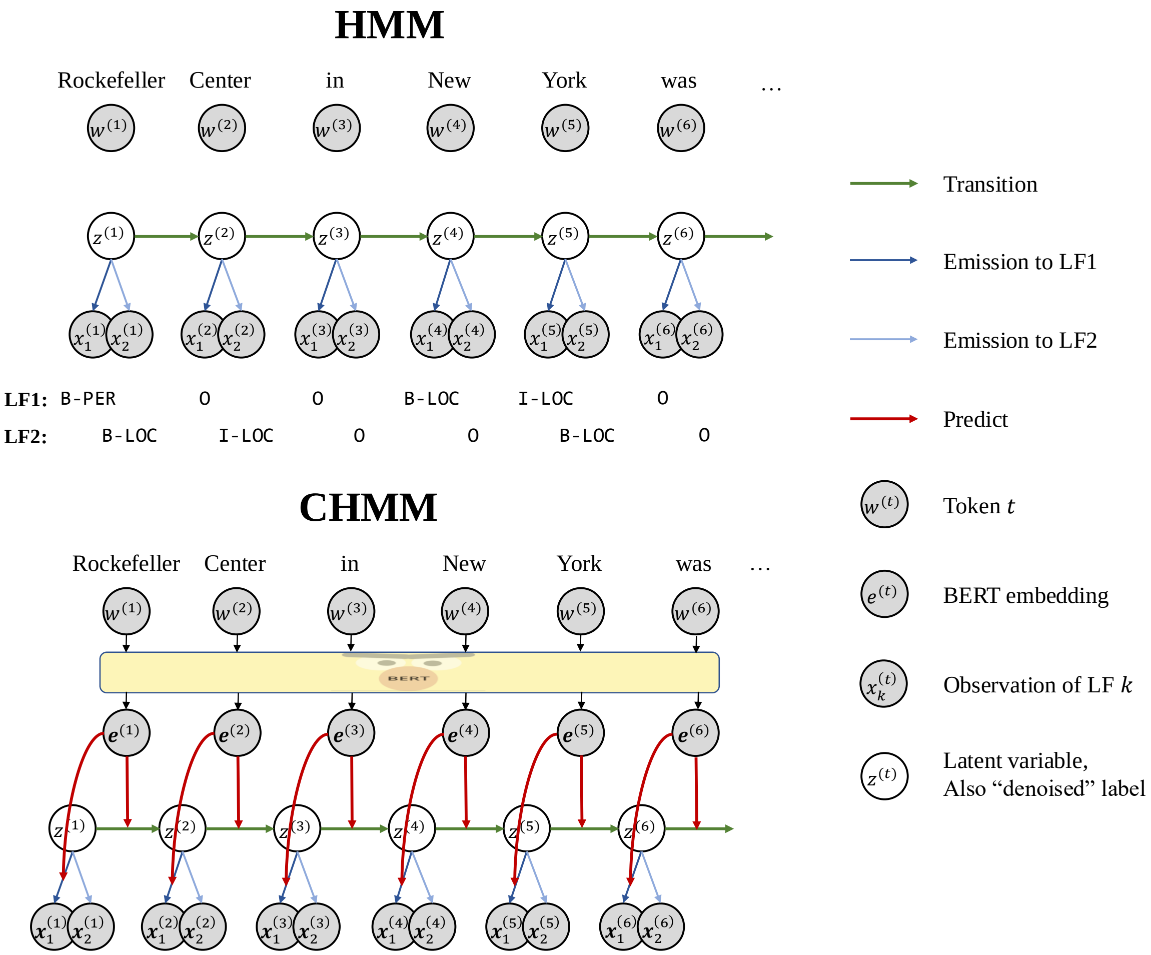

HMM
Rockefeller
Center
in
New
York
was
…
$w^{(1)}$ $w^{(2)}$ $w^{(3)}$ $w^{(4)}$ $w^{(5)}$ $w^{(6)}$
$z^{(1)}$ $z^{(2)}$ $z^{(3)}$ $z^{(4)}$ $z^{(5)}$ $z^{(6)}$
$x_1^{(1)}$ $x_2^{(1)}$ $x_1^{(2)}$ $x_2^{(2)}$ $x_1^{(3)}$ $x_2^{(3)}$ $x_1^{(4)}$ $x_2^{(4)}$ $x_1^{(5)}$ $x_2^{(5)}$ $x_1^{(6)}$ $x_2^{(6)}$
LF1: B-PER O O B-LOC I-LOC O
LF2: B-LOC I-LOC O O B-LOC O
CHMM
Rockefeller
Center
in
New
York
was
…
$w^{(1)}$ $w^{(2)}$ $w^{(3)}$ $w^{(4)}$ $w^{(5)}$ $w^{(6)}$
BERT
$e^{(1)}$ $e^{(2)}$ $e^{(3)}$ $e^{(4)}$ $e^{(5)}$ $e^{(6)}$
$z^{(1)}$ $z^{(2)}$ $z^{(3)}$ $z^{(4)}$ $z^{(5)}$ $z^{(6)}$
$x_1^{(1)}$ $x_2^{(1)}$ $x_1^{(2)}$ $x_2^{(2)}$ $x_1^{(3)}$ $x_2^{(3)}$ $x_1^{(4)}$ $x_2^{(4)}$ $x_1^{(5)}$ $x_2^{(5)}$ $x_1^{(6)}$ $x_2^{(6)}$
Transition
Emission to LF1
Emission to LF2
Predict
$w^{(t)}$ Token t
$e^{(t)}$ BERT embedding
$x_k^{(t)}$ Observation of LF k
$z^{(t)}$ Latent variable,
Also "denoised" label

Figure 4.5: Model details of HMM and CHMM. CHMM predicts the token-wise transition and emission probabilities while HMM uses fixed semantics-independent probabilities.

PLM embeddings $\boldsymbol{e}^{(1:T)}$ for the input tokens $\boldsymbol{w}^{(1:T)}$. This approach leverages the rich contextual information from PLM embeddings and overcomes the limitation of static matrices.

4.3.1 Overall Model Architecture

In CHMM, $\boldsymbol{\Psi}^{(t)} \in [0,1]^{L\times L}$ represents the transition matrix at time step t, which is the probability of transitioning from true label i to j: $\Psi_{i,j}^{(t)} \triangleq p(z^{(t)} = j | z^{(t-1)} = i, \boldsymbol{e}^{(t)}), \quad i, j \in \mathbb{N}_{[1,L]}$. The emission matrix for LF k, $\boldsymbol{\Phi}_k^{(t)} \in [0,1]^{L\times L}$, has elements representing the likelihood of LF k observing label j given the true hidden label i at time step t: $\Phi_{k,i,j}^{(t)} \triangleq p(x_{k,j}^{(t)} = 1 | z^{(t)} = i, \boldsymbol{e}^{(t)})$. $\boldsymbol{\Psi}^{(t)}$ and $\boldsymbol{\Phi}_{1:K}^{(t)}$ are predicted from $\boldsymbol{e}^{(t)}$ through Fully Connected (FC) layers and reshaped to meet the expected matrix dimensionality. To ensure these matrices represent proper probability distributions, the SoftMax function is applied along the last axis, affecting the output labels for $\boldsymbol{\Psi}^{(t)}$ and observations for $\boldsymbol{\Phi}^{(t)}$.

Examining the emission matrix $\boldsymbol{\Phi}^{(t)} \in [0,1]^{K\times L\times L}$, we notice that the scalability issues of CHMM arise as K and L increase. To predict $\boldsymbol{\Phi}^{(t)}$ from $\boldsymbol{e}^{(t)}$, the FC dimensionality could reach $768KL^2$. For instance, CoNLL 2003 dataset features approximately 20 LFs and 4 entity types, resulting in $\sim 1.24 \times 10^6$ FC parameters. The subsequent SoftMax layer, further adds to the computational burden, making model training both costly and complex due to the non-convex nature of the optimization problem.

Therefore, rather than directly estimating all elements, Sparse-CHMM computes LF reliabilities $\widetilde{\boldsymbol{A}} \in [0,1]^{K\times L}$ using an NN, and then extends these into the base prior $\boldsymbol{\Lambda} \in [0,1]^{K\times L\times L}$ via pre-defined heuristic functions. Utilizing $\widetilde{\boldsymbol{A}}$ and LF observations $\boldsymbol{x}$, the WXOR scores $\widetilde{\boldsymbol{W}} \in [0,1]^{K\times L\times L}$ are derived, indicating the likelihood of LFs providing incorrect labels. These scores are adjusted by a NN-predicted matrix $\boldsymbol{C} \in [0,1]^{K\times L}$, creating the add-on prior $\boldsymbol{\Delta}$. The emission matrix $\boldsymbol{\Phi}$ is then generated from a Dirichlet distribution, informed by $\boldsymbol{\Lambda}$ and $\boldsymbol{\Delta}$, to aid in model training (subsection 4.3.4). This process is depicted in Figure 4.6.

In contrast to transitions, emissions exhibit significantly less variability relative to input tokens. Therefore, we use sentence-level emissions from the sentence embedding $\boldsymbol{e}^{(0)}$.

4.3.2 Labeling Function Reliability and Emission Base Prior

For LF k, the emission diagonal elements $\Phi_{k,l,l} \triangleq p(x_{k,l}^{(t)} = 1 | z^{(t)} = l, \boldsymbol{e}^{(0)}),\ l \in \mathbb{N}_{[1,L]}$, as depicted in Figure 4.2, have a unique physical significance: they represent the probabilities of LF k correctly identifying the labels, which can be regarded as LF reliability. Conversely, a plausible emission matrix can be formulated based on the known performance of an LF.

Upon establishing this framework, we proceed to forecast the reliability logits utilizing the sentence embedding

$$\boldsymbol{A} \in \mathbb{R}^{K \times L} = \text{reshape}(\text{FC}(\boldsymbol{e}^{(0)})), \tag{4.1}$$

which, after being normalized along the label dimension, is integrated into the diagonal of $\boldsymbol{\Phi}$. Nonetheless, given that LFs more frequently encounter the label `O` (indexed as 1) than other labels, the model inclines to accentuate the corresponding weight $\Phi_{k,l,1}$ within the emission. Due to the constraint that $\boldsymbol{\Phi}_{k,l}$ sums to 1, an inflated $\Phi_{k,l,1}$ diminishes the diagonal values $\Phi_{k,l,l}$ excessively. This imbalance is rectifiable by employing SoftMax across the LFs:

$$\hat{\boldsymbol{A}}_{:,l} = \operatorname{softmax}(\boldsymbol{A}_{:,l}),\ l \geq 2; \quad \hat{\boldsymbol{A}}_{:,1} = \sigma(\boldsymbol{A}_{:,1}). \tag{4.2}$$

This ensures a high score for at least one LF, preventing the model from overlooking all weak annotations. Given that $\boldsymbol{A}_{k,1}$ signifies the emission confidence for `O`, it suffices to bound its value within $(0, 1)$ using the element-wise sigmoid function σ:

However, the SoftMax enforces $\sum_k \hat{\boldsymbol{A}}_{k,l} = 1$. This condition creates a *fixed-size* pool for allocating reliability scores, limiting model flexibility by not accommodating scenarios where either multiple LFs are confident or none are. As a compensation, we introduce a

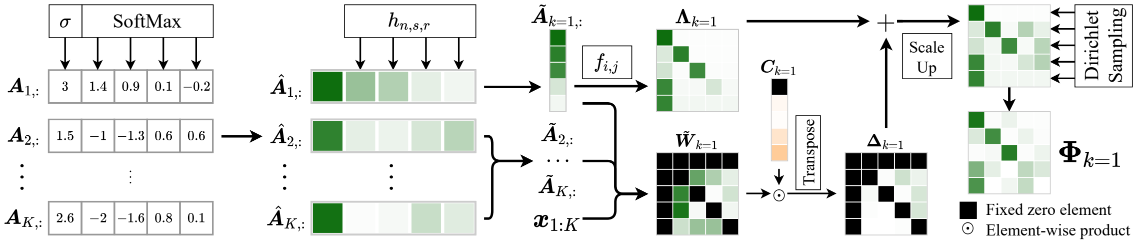

σ
SoftMax
$\boldsymbol{A}_{1,:}$
3
1.4
0.9
0.1
−0.2
$\boldsymbol{A}_{2,:}$
1.5
−1
−1.3
0.6
0.6
$\boldsymbol{A}_{K,:}$
2.6
−2
−1.6
0.8
0.1
$h_{n,s,r}$
$\hat{\boldsymbol{A}}_{1,:}$
$\hat{\boldsymbol{A}}_{2,:}$
$\hat{\boldsymbol{A}}_{K,:}$
$\tilde{\boldsymbol{A}}_{k=1,:}$
$f_{i,j}$
$\tilde{\boldsymbol{A}}_{2,:}$
$\tilde{\boldsymbol{A}}_{K,:}$
$\boldsymbol{x}_{1:K}$
$\boldsymbol{\Lambda}_{k=1}$
$\tilde{\boldsymbol{W}}_{k=1}$
$\boldsymbol{C}_{k=1}$
Transpose
$\boldsymbol{\Delta}_{k=1}$
Scale Up
Dirichlet Sampling
$\boldsymbol{\Phi}_{k=1}$
Fixed zero element
⊙ Element-wise product

Figure 4.6: Pipeline for constructing emissions for LF $k = 1$ across 5 labels. Darker shades represent higher values within the $[0, 1]$ interval.

piecewise scaling function $h_{n,s,r}(\cdot)$, defined as

$$\widetilde{A}_{k,l} = h_{n,s,r}(\hat{A}_{k,l});$$
$$h_{n,s,r}(a) = \begin{cases} \frac{1}{r^{n-1}} a & a^{\frac{1}{s}} < r; \\ -\frac{1}{(1-r)^{(n-1)}}(1 - a^{\frac{1}{s}})^n + 1 & a^{\frac{1}{s}} \geq r. \end{cases} \tag{4.3}$$

It calibrates the SoftMax output with scales defined by exponents n and s. To enable more subtle control of the scores, we utilize the split point $r \in [0, 1]$ to segment the input domain into upper and lower halves, each scaled differently.

Subsequently, we extend $\widetilde{\boldsymbol{A}}$ into the *emission base prior* matrix $\boldsymbol{\Lambda} \in [0, 1]^{K \times L \times L}$, employing expansion functions based on latent state $z^{(t)} = i$ and the observation $x_{k,j}^{(t)} = 1$:

$$\Lambda_{k,i,j} = f_{i,j}(\widetilde{A}_{k,i}); \quad \forall i, j \in 1 : L,$$
$$f_{i,j}(a) = \begin{cases} a & i = j; \\ \frac{1}{L-1}(1 - a) & i = 1, j \geq 2; \\ g_{n,r}(a) & i \geq 2, j = 1; \\ \frac{1}{L-2}(1 - a - g_{n,r}(a)) & i \geq 2, j \geq 2, i \neq j, \end{cases} \tag{4.4}$$

$$g_{n,r}(a) = \begin{cases} \frac{2-L}{(n-1)r^n - nr^{n-1}} a^n + (1 - L)x + 1 & a \leq r; \\ \frac{g_{n,r}(r)}{r-1} a - \frac{g_{n,r}(r)}{r-1} & a > r, \end{cases} \tag{4.5}$$

where n is the exponential term and r is the split point.

LF k's reliability scores $\widetilde{\boldsymbol{A}}_k$ populate the diagonal of $\boldsymbol{\Lambda}_k$ (*i.e.*, $\boldsymbol{\Lambda}_{k,l,l} = \widetilde{\boldsymbol{A}}_{k,l}$). For latent state O ($z^{(t)} = i = 1$), emissions to non-O labels, $p(\boldsymbol{x}_{k,j}^{(t)} = 1 | z^{(t)} = 1, \boldsymbol{e}^{(0)}),\ \forall j \geq 2$, are uniformly distributed, summing to $1 - \widetilde{A}_{k,1}$. For latent states other than O ($z^{(t)} = i \geq 2$), the likelihood of observing O surpasses that of other entities, resulting in higher and more pivotal emit-to-O probabilities $p(\boldsymbol{x}_{k,1}^{(t)} = 1 | z^{(t)} = i, \boldsymbol{e}^{(0)}),\ \forall i \geq 2$, as illustrated in Figure 4.2. Low reliability scores indicate an LF's indecisiveness, leading to uniformly distributed off-diagonal non-O values near the diagonal values. Above some threshold r, an LF is deemed sufficiently confident, reducing the emission probabilities to other entities. The exponent n determines the rate at which emit-to-O probabilities decline, influencing how closely off-diagonal values approach the diagonal before reaching the threshold r. Please refer to [60] for further details.

4.3.3 WXOR and Emission Add-on Prior

The off-diagonal values $\Phi_{k,i,j}; i, j \geq 2; i \neq j$ signify the likelihood of LF k erroneously classifying label i as j. These probabilities are typically low, yet they can be significant in certain instances, as demonstrated in Figure 4.2, which cannot be addressed by the base prior $\boldsymbol{\Lambda}$.

To better represent misclassification probabilities, we introduce WXOR scores $\widetilde{\boldsymbol{W}} \in [0,1]^{K \times L \times L}$. For LF k relative to other LFs $k' \in \mathbb{N}_{[1,K]\backslash k}$, we define a WXOR logit between a query label $l^{\text{query}} \in \mathbb{N}_{[2,L]}$ and a target label $l^{\text{tgt}} \in \mathbb{N}_{[2,L]}$:

$$\begin{gathered} W_{k,l^{\text{query}},l^{\text{tgt}}}^{(t)} = (1 - \widetilde{A}_{k,l^{\text{query}}}) x_{k,l^{\text{query}}}^{(t)} \sum_{k'=1}^{K} \widetilde{A}_{k',l^{\text{tgt}}} x_{k',l^{\text{tgt}}}^{(t)}; \\ \boldsymbol{W}_{k,1,:}^{(t)} = \boldsymbol{W}_{k,:,1}^{(t)} = \boldsymbol{0}; \quad W_{k,l,l}^{(t)} = 0,\ l \in \mathbb{N}_{[1,L]}. \end{gathered} \tag{4.6}$$

WXOR quantifies the error probability of LF k for label l^{query} against other LFs' confidence in l^{tgt}. A high $W_{k,l^{\text{query}},l^{\text{tgt}}}^{(t)}$ indicates LF k's uncertainty in observing l^{query} juxtaposed with other LFs' confidence in l^{tgt}. $W^{(t)}_{k,l^{\text{query}},l^{\text{tgt}}}$ is nonzero only if LF k uniquely observes l^{query}, analogizing an XOR gate. Labels O and diagonal entries do not require WXOR. With Equation 4.6, we aggregate $\boldsymbol{W}^{(t)}$ tensors across M sentences, each with length T_m, to compile the aggregate WXOR:

$$\hat{W}_{k,l^{\text{query}},l^{\text{tgt}}} = \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} W^{(t)}_{m,k,l^{\text{query}},l^{\text{tgt}}}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} x^{(t)}_{m,k,l^{\text{query}}}}. \tag{4.7}$$

Directly adding $\hat{\boldsymbol{W}}$ into $\boldsymbol{\Lambda}$ is impractical due to unknown scale ratios. As a solution, we employ a scaling factor

$$\boldsymbol{C} \in (0,1)^{K\times L} = \text{reshape}(\sigma(\text{FC}(\boldsymbol{e}^{(0)}))). \tag{4.8}$$

Since $\boldsymbol{C}$ can only downscale values, we adjust $\hat{\boldsymbol{W}}$ elements to ensure sufficiently large pre-scaling values $\widetilde{\boldsymbol{W}}_{k,:,l^{\text{tgt}}} = \operatorname{softmax}(\hat{\boldsymbol{W}}_{k,:,l^{\text{tgt}}})$. Subsequently, the *emission addon prior* $\boldsymbol{\Delta} \in [0,1]^{K\times L\times L}$ is constructed by rescaling $\widetilde{\boldsymbol{W}}_k$ rows:

$$\boldsymbol{\Delta}_{k,:,l} = C_{k,l} \times \widetilde{\boldsymbol{W}}_{k,l,:}. \tag{4.9}$$

Notice that in $\boldsymbol{\Delta}_k$, the target labels are at the first axis and the query labels second, different from $\widetilde{\boldsymbol{W}}_k$. As the target labels are essentially the latent states we want to predict, this unifies the physical meaning of $\widetilde{\boldsymbol{W}}$ and the emission matrix $\boldsymbol{\Phi}$.

4.3.4 Emission Matrix Sampling

We opt to sample the latent emission variables from the Dirichlet distribution parameterized by the base and addon priors $\boldsymbol{\Lambda}$ and $\boldsymbol{\Delta}$, which is more robust and avoids the saddle points. Moreover, Dirichlet distribution does not post sum-up-to-one constraints to its parameters, granting greater flexibility in parameter selection. We formulate the concentration parameters $\boldsymbol{\Omega}$ by summing the base and addon priors and then applying a scale

$$\boldsymbol{\Omega} \in \mathbb{R}_{+}^{K \times L \times L} = \nu^{\text{expan}} \times (\boldsymbol{\Lambda} + \boldsymbol{\Delta}) + \nu^{\text{base}}, \tag{4.10}$$

where $\nu^{\text{base}} \in \mathbb{R}_{+}$ and $\nu^{\text{expan}} \in \mathbb{R}_{+}$ are designed to regulate the baseline concentration and the span of concentration, respectively. The emission matrix $\boldsymbol{\Phi}_k$ for LF k is then sampled row-wise from this distribution:

$$\boldsymbol{\Phi}_{k,l} \sim \text{Dir}(\boldsymbol{\Omega}_{k,l}). \tag{4.11}$$

We employ pathwise derivative estimators [151] to propagate the gradient through the Dirichlet sampling.

Dirichlet sampling is only applied when network layers requires backpropagation. On other occasions, such as validation and test, the samples are substituted by the mean of the Dirichlet distribution.

4.3.5 Model Initialization

HMM typically initializes transition and emission probabilities with statistical estimates $\boldsymbol{\Psi}^*$ and $\boldsymbol{\Phi}^*$ derived from observations. However, there is no direct correspondence between the estimated matrix initials and NN parameters. To circumvent this, we pre-train CHMM and Sparse-CHMM by minimizing the Mean Squared Error (MSE) loss ℓ_{MSE} between their outputs and the target statistics: $\ell_{\text{MSE}} = \frac{1}{T} \sum_t \|\boldsymbol{\Psi}^* - \boldsymbol{S}^{(t)}\|_F^2 + \|\boldsymbol{\Phi}^* - \boldsymbol{H}^{(t)}\|_F^2$. $\|\cdot\|_F$ denotes the Frobenius norm, and $\boldsymbol{S}^{(t)}$ and $\boldsymbol{H}^{(t)}$ are the predicted transition and emission matrices at time step t, respectively. The model is trained on the training set, and the statistics are computed from the training set as well.

4.3.6 Inference

Upon completion of training, CHMM and Sparse-CHMM are capable of determining the most likely sequence of hidden labels $\hat{\boldsymbol{z}}^{(1:T)}$:

$$\hat{\boldsymbol{z}}^{(1:T)} = \arg\max_{\boldsymbol{z}^{(1:T)}} p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z}^{(1:T)} | \boldsymbol{x}_{1:K}^{(1:T)}, \boldsymbol{e}^{(0:T)}), \tag{4.12}$$

with $\hat{\boldsymbol{\theta}}$ denoting the model's trained parameters. These outcomes are obtainable through the Viterbi decoding algorithm or by maximizing the smoothed marginal $\boldsymbol{\gamma}^{(1:T)}$.

4.3.7 Training Objective

According to the generative process of CHMM, the joint distribution of hidden states and observed weak labels for a sequence, $p(\boldsymbol{z}^{(0:T)}, \boldsymbol{x}^{(1:T)} | \boldsymbol{\theta})$, is decomposable as follows:

$$p(\boldsymbol{z}^{(0:T)}, \boldsymbol{x}^{(1:T)} | \boldsymbol{\theta}) = p(z^{(0)}) p(\boldsymbol{x}^{(1:T)} | \boldsymbol{z}^{(1:T)}) = p(z^{(0)}) \prod_{t=1}^{T} p(z^{(t)} | z^{(t-1)}) \prod_{t=1}^{T} p(\boldsymbol{x}^{(t)} | z^{(t)}), \tag{4.13}$$

where $\boldsymbol{\theta}$ symbolizes all trainable parameters. HMMs are typically trained using an Expectation-Maximization (EM) algorithm. During the E-step, we calculate the expected complete data log likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{i=1}^{L} \gamma_i^{(0)} \log \pi_i + \sum_{t=1}^{T} \sum_{i=1}^{L} \sum_{j=1}^{L} \xi_{i,j}^{(t)} \log \Psi_{i,j}^{(t)} + \sum_{t=1}^{T} \sum_{i=1}^{L} \gamma_i^{(t)} \log \varphi_i^{(t)}, \tag{4.14}$$

where $\varphi_i^{(t)} \triangleq p(\boldsymbol{x}^{(t)} | z^{(t)} = i) = \prod_{k=1}^{K} \sum_{j=1}^{L} \Phi_{k,i,j}^{(t)} x_{k,j}^{(t)}$ is the observation likelihood; $\pi_1 = 1, \boldsymbol{\pi}_{2:L} = \boldsymbol{0}$ represents the initial hidden status; $\gamma_i^{(t)} \triangleq p(z^{(t)} = i | \boldsymbol{x}^{(1:T)})$ is the smoothed marginal; $\xi_{i,j}^{(t)} \triangleq p(z^{(t-1)} = i, z^{(t)} = j | \boldsymbol{x}^{(1:T)})$ signifies the expected number of transitions. These variables are derived using the forward-backward algorithm, which is detailed in section C.1.

In the M-step, we update the model parameters by maximizing Equation 4.14 through

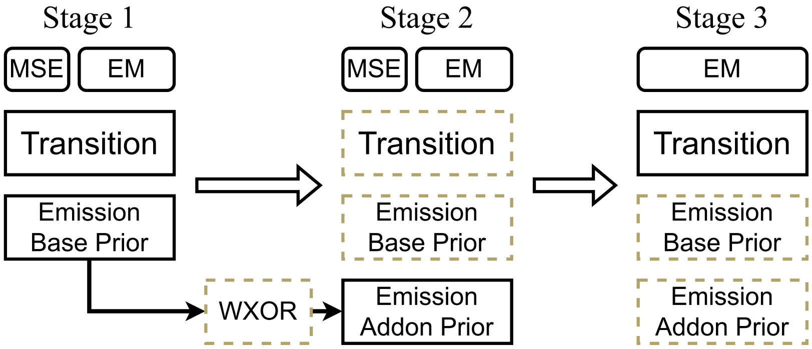


Figure 4.7: Sparse-CHMM training procedure. The model parameters of the dotted squares are frozen. MSE and EM are optimization approaches associated with the model pre-training and training steps.

gradient ascent $\nabla\boldsymbol{\theta} = \frac{\partial Q(\boldsymbol{\theta},\boldsymbol{\theta}^{\text{old}})}{\partial\boldsymbol{\theta}}$. This is implemented through backpropagation, and the details are provided in the original paper [59].

4.3.8 Training Stages

As the computation of the emission is complicated and inter-dependent, we adopt a three-stage training strategy to allow sufficient training of each model component, as shown in Figure 4.7. It decouples the optimization of the model components while keeping the training efficient.

Stage 1 is focused on optimizing the transition matrix $\boldsymbol{\Psi}$ and the emission base prior $\boldsymbol{\Lambda}$, excluding the addon prior by setting $\boldsymbol{\Delta} = \mathbf{0}$. The reason is straightforward: calculating the WXOR scores $\widetilde{\boldsymbol{W}}$ requires high-quality reliability scores $\widetilde{\boldsymbol{A}}$, which are obtained by optimizing $\boldsymbol{\Lambda}$. Stage 2 trains the addon prior $\boldsymbol{\Delta}$, leaving $\boldsymbol{\Psi}$ and $\boldsymbol{\Lambda}$ frozen. This stage aims to search for the best scaling factors $\boldsymbol{C}$ for $\widetilde{\boldsymbol{W}}$, and freezing the irrelevant parameters relieves the training pressure. $\widetilde{\boldsymbol{W}}$ is calculated right after stage 1 with all training and validation instances and remains constant for stages 2 and 3.

Stage 3 is inspired by our empirical discovery about HMMs. We find that if we only optimize the transition matrix $\boldsymbol{\Psi}$ with the emission fixed, generally to the true values, the model performance can be improved. The true emission is the statistics of the annotations

$\boldsymbol{x}_k$ of LF k given ground-truth labels $\boldsymbol{y}$:

$$\Phi^{\text{true}}_{k,i,j} = p(\boldsymbol{x}_k|\boldsymbol{y}) \triangleq \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\mathbb{I}(x^{(t)}_{m,k,j}=1, y^{(t)}_m = i)}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\mathbb{I}(y^{(t)}_m = i)},$$

where $\mathbb{I}(\cdot)$ is the indicator function. Consequently, we believe that continuing training the transition of Sparse-CHMM for several more epochs with the emission frozen would lead to a similar performance improvement.

In addition, stages 1, 2, and 3 use different pre-training strategies. The pre-training of stage 1 is the same as subsection 4.3.5. Stage 2's pre-training is to initialize the NN parameters associated with the addon prior $\boldsymbol{\Delta}$ only, so we drop the transition part of the MSE loss and substitute the target emission by $\boldsymbol{\Phi}' = \lambda\boldsymbol{\Phi}^* + \frac{(1-\lambda)}{M}\sum_{m=1}^{M}\boldsymbol{\Phi}^{(I)}_m$ to take advantage of stage 1's results. Here $\lambda \in [0,1]$ is the weight of observation statistics, which is fixed to 0.2 in our experiments. $\boldsymbol{\Phi}^{(I)}_m$ is the optimized emission from stage 1. Stage 3 successes all model parameters from the previous stage and thus has no pre-training.

Another issue with the training is the LF competition. When several LFs classify different tokens as the same entity, they have to compete for the reliability score due to the normalization brought by the SoftMax function . In this case, the LF with the highest observation frequency tends to dominate the score, making other LFs neglected. To deal with this issue, we append a simple MV to the LF set during the training of the reliability scores. MV aggregates the token-entity mappings from all LFs into a single annotation sequence, which forces the model to pay equal attention to the patterns and resolves the competition.

4.3.9 Complexity Analysis

Compared with CHMM, Sparse-CHMM significantly reduces the training resource consumption by predicting sparse emission elements. The emission NN parameter number and computation complexity are shown in Table 4.1, where the transition attributes are not presented because they are the same for both models. The factor 2 for Sparse-CHMM comes

Table 4.1: Emission complexity of each epoch. M is the number of training sentences, T is the average of sentence lengths, and d^{emb} is the dimension of the PLM embeddings.

	CHMM	Sparse-CHMM
# Emission Parameters	$d^{\text{emb}} \times K \times L^2$	$2 \times d^{\text{emb}} \times K \times L$
Emission Complexity	$\mathcal{O}(M \times T \times d^{\text{emb}} \times K \times L^2)$	$\mathcal{O}(M \times d^{\text{emb}} \times K \times L)$

from matrices $\widetilde{\boldsymbol{A}}$ and $\boldsymbol{C}$. We can see that Sparse-CHMM reduces the emission NN parameter number to $2/L$ of CHMM and the complexity to $1/(T \times L)$, which are substantial when the number of entity labels L is large. The complexity of other emission elements is negligible because they do not contain the embedding dimension d^{emb}, which is much larger than other terms. Calculating the WXOR scores can be as complex as $\mathcal{O}(M \times T \times K^2 \times L^2)$, but they are calculated only once at stage 2 and stay fixed henceforth.

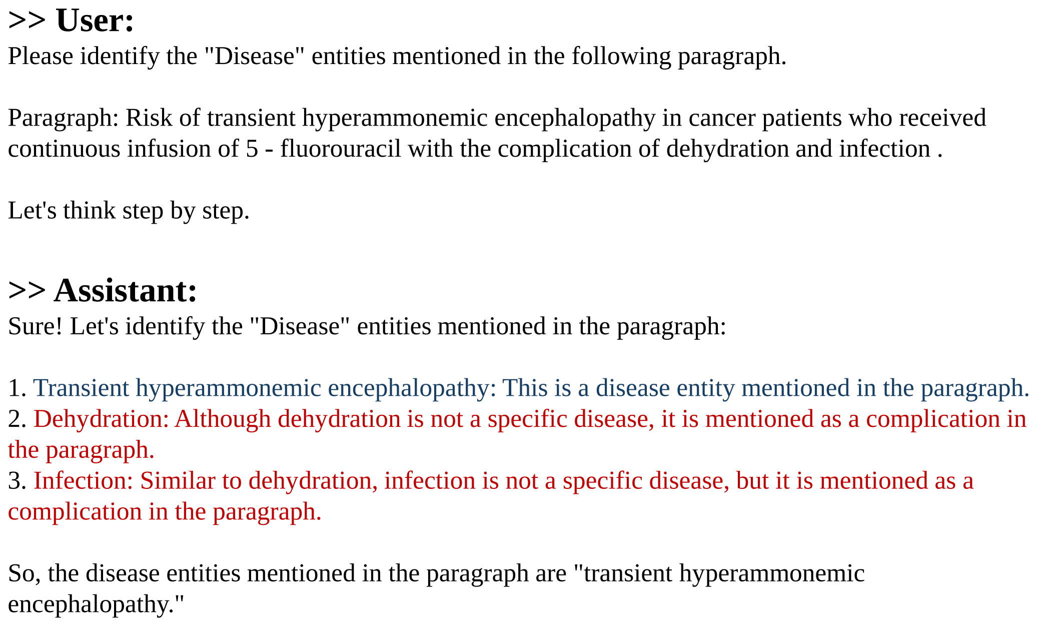
4.4 Autoregressive Information Extraction

Along the other line of research, to improve the zero-shot IE capabilities of LLMs when structured outputs are required, our prompting pipeline, G&O, introduces three main components, depicted in Figure 4.3:

- **Free-form response generation**, in which the LLMs are prompted to identify relevant information from the input context without imposing any structural or syntactic constraints;
- **Answer clean-up**, a task-dependent filtering step that removes superfluous or unrelated information to preserve the integrity of the final structured output; and
- **Structure organization**, which transforms the refined content into a well-defined format, such as a Markdown table or list, based on the LLMs' response history.

Additionally, we incorporate zero-shot CoT [152] to further enhance IE performance.

Although this design may seem like a minor adjustment compared to conventional IE prompts, which typically merge steps 1) free-form generation and 3) structure organization into a single prompt (Figure 4.3), it more closely aligns with the inherent flow of natural



>> User:

Please identify the "Disease" entities mentioned in the following paragraph.

Paragraph: Risk of transient hyperammonemic encephalopathy in cancer patients who received continuous infusion of 5 - fluorouracil with the complication of dehydration and infection .

Let's think step by step.

>> Assistant:

Sure! Let's identify the "Disease" entities mentioned in the paragraph:

1. Transient hyperammonemic encephalopathy: This is a disease entity mentioned in the paragraph.
2. Dehydration: Although dehydration is not a specific disease, it is mentioned as a complication in the paragraph.
3. Infection: Similar to dehydration, infection is not a specific disease, but it is mentioned as a complication in the paragraph.

So, the disease entities mentioned in the paragraph are "transient hyperammonemic encephalopathy."

Figure 4.8: GPT-3.5's natural language responses tend to include irrelevant entities (marked by red). Although clearly explained, irrelevant terms still pose a difficulty for GPT-3.5 during format organization.

language and often yields more coherent, informative responses [153].

Moreover, the clean-up stage is indispensable for ensuring the clarity of the final results. Figure 4.8 shows that while LLMs generally identify relevant entities or relationships, they also tend to include extraneous details unrelated to the requested types. This tendency likely stems from the models' training to be "helpful" through RLHF [22]. Despite having identified the correct entities, these additional details can complicate the subsequent organization of useful information into the desired format. Hence, an explicit clean-up phase is crucial for delivering concise, task-focused outputs. Finally, we adopt Markdown tables for the structured output due to their broad occurrence in LLM training data and for consistency with the RE pipeline.

Table 4.2: NER dataset statistics.

	CoNLL03 [154]	NCBI [155]	BC5CDR [133]	Laptop [136]	OntoNotes [135]	PolyIE [52]
# Instance	22,137	793	1,500	3,845	143,709	1,170
# Training	14,041	593	500	2,436	115,812	-
# Validation	3,250	100	500	609	5,000	-
# Test	3,453	100	500	800	22,897	1,170
# Entities	4	1	2	1	18	3
# LFs	16	5	9	3	17	-

4.5 Experiment Setup

4.5.1 Weakly Supervised Named Entity Recognition

Datasets We consider 5 NER datasets that span generic to highly specialized domains, thus providing a broad basis for evaluating weakly supervised NER methods. Specifically:

- CoNLL 2003 (English subset) [154] consists of 22,137 sentences from Reuters news stories, annotated with 4 entity types: PER, LOC, ORG, and MISC.
- LaptopReview [136] comprises 3,845 sentences of laptop reviews. All laptop-related Terms are treated as named entities.
- NCBI-Disease [155] contains 793 PubMed abstracts, each annotated with Disease mentions.
- BC5CDR [133] consists of 1,500 PubMed articles annotated with Chemical and Disease entities.
- OntoNotes 5.0 [135] is a large-scale dataset of 143,709 sentences labeled with 18 fine-grained entity types.

For all datasets, we use the LFs provided by the Wrench benchmark platform [49], which are curated for weakly supervised NER. Table 4.2 reports the number of entities and labeling functions for each dataset.

Baselines We compare our proposed model with representative weakly supervised NER baselines from the Wrench benchmark:

- MV simply selects the majority vote among all LFs, resolving ties randomly.
- Snorkel [56] is a context-free, token-based graphical model that treats tokens independently.
- HMM [138, 57, 139] is a popular sequence model for weakly supervised NER, offering limited context modeling via Markov assumptions.
- CHMM [59] extends HMM by predicting token-wise transition and emission probabilities from BERT embeddings through NNs.
- ConNet [141] exploits a context-aware attention mechanism over Conditional Random Field (CRF) representations produced by different LFs.

Additionally, we include 3 supervised reference methods:

- A fully supervised BERT-NER model, trained directly on human annotations.
- The best consensus of LFs, serving as an oracle that always selects the correct token labels whenever any LF provides them.
- CHMM-FE, which is the CHMM model augmented with fixed ground-truth emission probabilities (see section 4.3.8).

Note that each supervised reference has direct or partial access to the true labels during training, differentiating them from the purely weakly supervised baselines.

Evaluation Metrics We evaluate all NER label models using micro-averaged, entity-level precision, recall, and F$_1$ scores. Specifically, let TP, FP, and FN denote the total counts of true positives, false positives, and false negatives, respectively, aggregated over all entities and classes. Then, we compute the micro-averaged precision, recall, and F$_1$ as follows:

$$\begin{gathered}\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.\end{gathered} \tag{4.15}$$

These metrics are computed via the `seqeval` package in Python, which evaluates entity boundaries at the micro (global) level. In other words, a predicted entity is considered correct only if its boundaries exactly match those of the ground-truth entity. Unless otherwise specified, we report the average performance over 5 random trials to reduce variance.

Implementation Details We adopt different PLMs, specifically, BERT variants, across datasets to reflect common usage for each domain:

- `bert-base-uncased` [10] on CoNLL 2003, LaptopReview, and OntoNotes 5.0,
- `bioBERT` [156] on NCBI-Disease,
- `SciBERT` [157] on BC5CDR.

Although we do not specifically compare the effectiveness of various language models, different domain-specific PLMs may yield further performance variations.

As we proposed in [49], we adopt an inductive evaluation: each model is trained on the training set, validated on the development set, and evaluated on the test set. We use the validation set for hyper-parameter tuning and early stopping. Moreover, we calculate the WXOR scores on the union of the training and validation sets to ensure consistency with prior work.

4.5.2 Autoregressive Zero-Shot Information Extraction

For the sake of clarity, we focus on the zero-shot NER subtask in this section. Readers interested in zero-shot RE are referred to the G&O paper [61] for a detailed discussion and experimental results.

Datasets To evaluate the zero-shot performance of G&O, we adopt 4 NER datasets covering both general and domain-specific corpora, thus providing a diverse test bed. Apart from the aforementioned CoNLL 2003, NCBI-Disease, and BC5CDR datasets, we also include:

- PolyIE [52] is a newly introduced materials science dataset. It consists of 96 paragraphs and 1,170 sentences, with 2,761 and 188 tokens, respectively. These passages are annotated with three entity types: `material name`, `property name`, and `property value`.

Since our goal is to test zero-shot capabilities, we use only the test sets of these corpora, providing no task-specific fine-tuning data to the model.

Baselines We benchmark G&O against two autoregressive prompting methods:

- One-Step prompting consolidates all extraction steps into a single instruction, such as "Identify all entities of the following types in the given text." While specific prompts vary, this one-step approach is commonly used in LLMs to elicit structured outputs. It often incorporates a secondary "clean-up" phase to refine the extracted entities.
- All-Entity-in-One (AEiO) instructs the model to identify multiple entity types simultaneously (*e.g.*, "Identify `person`, `location`, and `organization` entities in the following paragraph."). This differs from G&O primarily in how many entity types are addressed at once.

Compared to these baselines, G&O employs a more structured, multi-step strategy, potentially improving consistency and accuracy in extracting entities across different domains.

Metrics and Evaluation We report micro-averaged precision, recall, and F_1 to quantify system performance. However, unlike the previous settings, zero-shot extraction with autoregressive LLMs requires additional post-processing to map the model's semi-structured responses to specific entity spans in the source text. To mitigate errors caused by format variations (*e.g.*, additional spaces or rephrasings), we apply a fuzzy matching algorithm from Python's `difflib` library to align predicted entity strings with the original text.

Following [147], we evaluate under both **full** and **partial** span matching criteria:

- Full: The predicted span must exactly match the ground-truth entity boundaries.
- Partial: Any overlap between the predicted and ground-truth spans is counted as a

Table 4.3: Weakly Supervised NER results, presented as “F_1 (Precision / Recall)” in %.

Models	CoNLL 2003	NCBI-Disease	BC5CDR	LaptopReview	OntoNotes 5.0
BERT-NER	90.74 (90.37 / 91.10)	88.89 (87.05 / 90.82)	88.81 (87.12 / 90.57)	81.34 (82.02 / 80.67)	84.11 (83.11 / 85.14)
Best consensus	86.73 (98.62 / 77.39)	81.65 (99.85 / 69.06)	88.42 (99.86 / 79.33)	77.60 (100.0 / 63.40)	85.11 (97.35 / 75.61)
CHMM-FE	71.43 (72.89 / 70.02)	81.86 (90.75 / 74.55)	86.45 (91.73 / 81.75)	72.38 (88.13 / 61.41)	67.99 (65.23 / 71.00)
ConNet	66.02 (67.98 / 64.19)	63.04 (74.55 / 55.16)	72.04 (77.71 / 67.18)	50.36 (63.04 / 42.73)	60.58 (59.43 / 61.83)
MV	60.36 (59.06 / 61.72)	78.44 (93.04 / 67.79)	80.73 (83.79 / 77.88)	73.27 (88.86 / 62.33)	58.85 (54.17 / 64.40)
Snorkel	62.43 (61.62 / 63.26)	78.44 (93.04 / 67.79)	83.50 (91.69 / 76.65)	73.27 (88.86 / 62.33)	61.85 (57.44 / 66.99)
HMM	62.18 (66.42 / 58.45)	66.80 (**96.79** / 51.00)	71.57 (**93.48** / 57.98)	73.63 (89.30 / 62.63)	55.67 (57.95 / 53.57)
CHMM	63.22 (61.93 / 64.56)	78.74 (93.21 / 68.15)	83.66 (91.76 / 76.87)	73.26 (88.79 / 62.36)	64.06 (59.70 / **69.09**)
Sparse-CHMM	**71.53** (**73.80** / **69.39**)	**82.24** (93.18 / **73.60**)	**86.63** (89.56 / **83.88**)	**75.90** (**91.94** / **64.62**)	**64.85** (**61.26** / 68.88)

correct prediction, allowing minor discrepancies (*e.g.*, added or missing words). For example, in the sentence “He’s working for the White House,” if the ground truth labels “White House → `Organization`,” a predicted span of “the White House → `Organization`” counts as a correct entity for partial matching but is penalized under full matching due to the extra word “the” In both matching regimes, an incorrect type (*e.g.*, labeling “White House” as `Location`) is considered incorrect.

Overall, the combination of partial and full match metrics, along with fuzzy matching, provides a comprehensive view of model performance in challenging zero-shot settings.

4.6 Results and Analysis for Weakly Supervised Named Entity Recognition

4.6.1 Main Results

Table 4.3 presents a comparative evaluation of CHMM and Sparse-CHMM against baseline models on the Wrench benchmark [49]. Notably, Sparse-CHMM outperforms all competing label models, achieving an average F_1 score improvement of 3.01 points compared to the strongest baseline. The performance gain of Sparse-CHMM predominantly stems from an improvement in recall. Sparse-CHMM exhibits higher recall than even the best consensus on 3 out of 5 datasets. This substantial enhancement can be attributed to the robust emission structure of Sparse-CHMM, which effectively preserves correct annotations provided even by relatively weak LFs. This superior recall indicates that Sparse-CHMM can successfully identify entities that are not explicitly observed by *any* LF, an ability enabled

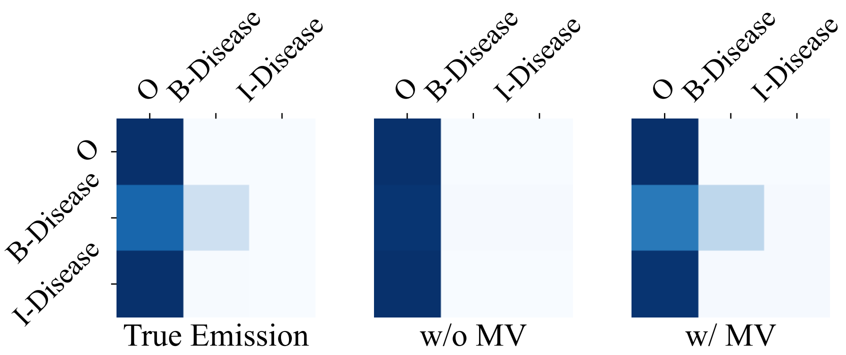


Figure 4.9: Emissions of LF "tag-CoreDictionaryExact" trained on NCBI-Disease with-/without the additional MV LF.

by the precisely estimated token-wise transition probabilities within the model.

Interestingly, Sparse-CHMM consistently surpasses the performance of CHMM-FE, despite the latter having access to ground-truth labels. This phenomenon occurs because using ground-truth labels directly to construct emission matrices in CHMM-FE may not always yield optimal model parameters. Moreover, the static emission matrices in CHMM-FE disregard sentence-specific variations, thereby limiting the model's ability to adaptively handle contextual nuances. In contrast, Sparse-CHMM dynamically adjusts emission probabilities according to the input sentences, better accommodating diverse linguistic patterns.

However, performance gaps between Sparse-CHMM and the best consensus are still observable on more challenging datasets such as CoNLL 2003 and OntoNotes 5.0. Notably, even the CHMM-FE model struggles to achieve high performance on these benchmarks. Beyond the inherent complexity due to the increased variety of entities and LFs, the primary challenge arises from the quality of the LFs themselves. Many LFs on these datasets have low recall, resulting in training dominated by the limited number of well-performing LFs. Consequently, the model's ability to leverage accurate annotations from these higher-quality LFs is compromised.

Table 4.4: Ablation F_1 scores.

	CoNLL	NCBI	BC5CDR	Laptop	OntoNotes
Sparse-CHMM	**71.53**	**82.24**	**86.63**	**75.90**	64.85
Training Stages					
Sparse-CHMM S1	69.73	77.89	86.15	73.59	64.15
Sparse-CHMM S2	70.79	79.18	86.04	74.42	**64.92**
Model Components					
Naïve emiss	33.02	77.32	-*	71.77	-*
w/o h_{nsr}	58.66	-*	86.48	75.38	64.69
w/o SoftMax	62.86	80.61	82.56	73.71	53.19
w/o Dirichlet	64.70	81.34	86.05	73.51	64.64
w/o S2	71.18	81.48	86.02	73.11	63.90
Merge S2 & S3	71.27	79.81	85.95	71.49	64.22

* The model fails training; the output labels are all "O".

4.6.2 Training Stages

Table 4.4 summarizes the test performance across different training stages. Overall, these results affirm the effectiveness of the proposed three-stage training strategy in enhancing Sparse-CHMM's performance. On BC5CDR, although a slight performance degradation occurs in stage 2, the additional priors established during this stage are crucial for the successful training of stage 3. Conversely, for OntoNotes 5.0, the modest performance drop observed in stage 3 can be attributed to discrepancies between the training and test datasets. Nevertheless, the consistent increase in validation F_1 scores from stage 2 to stage 3 underscores the robustness of the proposed training stages.

4.6.3 Model Components

To assess the impact of individual model components, we conduct extensive ablation studies involving multiple variants of Sparse-CHMM:

1. Sparse-CHMM with **naïve emission** matrices, where the reliability expansion function in Equation 4.5 is replaced by a uniform distribution across all non-diagonal elements, thereby diminishing emphasis on emit-to-O probabilities;

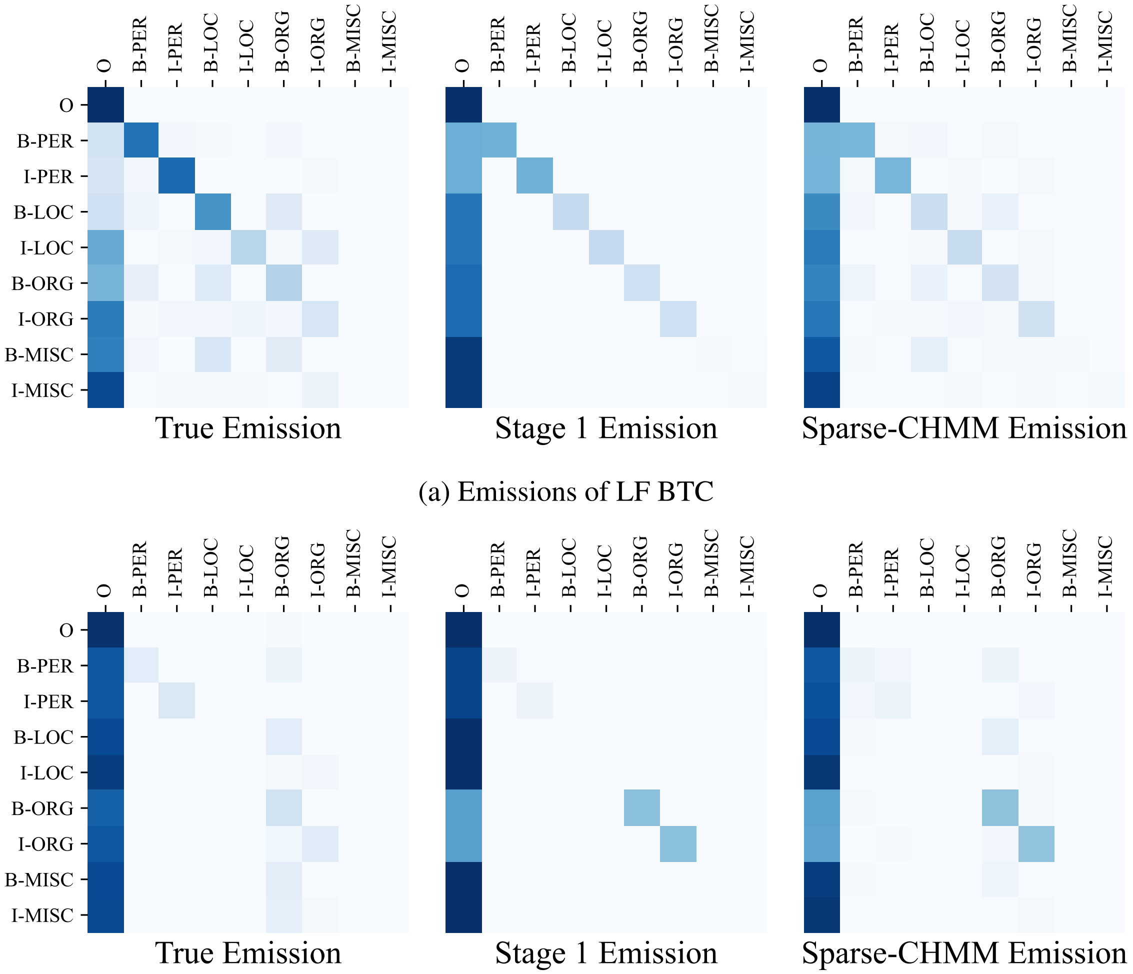
2. Sparse-CHMM **w/o** the scaling function $h_{n,s,r}$, equivalent to either replacing Equation 4.3 with $\tilde{\boldsymbol{A}} = \hat{\boldsymbol{A}}$ or setting the exponents $n = s = 1$;
3. Sparse-CHMM **w/o** the **SoftMax** function defined in Equation 4.2;
4. Sparse-CHMM **w/o** the **Dirichlet** sampling process during training;
5. Sparse-CHMM **w/o stage 2** training or omitting the WXOR scores;
6. Sparse-CHMM trained with **merged stages 2 and 3**, meaning the transition parameters remain unfrozen during stage 2.

Results presented in Table 4.4 clearly demonstrate that each individual component introduced in subsection 4.3.1 significantly contributes to model performance. The naïve emission variant exhibits markedly poor performance, underscoring the necessity of the carefully designed emission structure specified by Equation 4.4. Furthermore, the presence of both SoftMax and the scaling function $h_{n,s,r}$ is essential for accurately predicting reliability scores. Omitting Dirichlet sampling results in a greater susceptibility to local optima, hindering optimal convergence. Additionally, independently training each model component through the sequential stages achieves notably better outcomes compared to end-to-end training, highlighting the stability and effectiveness of the three-stage approach.

Figure 4.9 illustrates the critical role of incorporating the additional majority voting LF. Without this MV LF, the model exhibits bias against certain presented LFs, particularly failing to accurately assign reliability scores to observations labeled as `B-Disease`, thus significantly reducing Sparse-CHMM's F_1 to 66.48. Conversely, introducing the MV LF stabilizes reliability estimates, strengthens the emission structure, and significantly improves the F_1 to 82.24. Hence, the designed MV LF is essential for achieving optimal performance on specific datasets.

4.6.4 Case Studies

Figure 4.10 provides a comparison between the emission matrices predicted by Sparse-CHMM and the ground-truth matrices. Initially, Sparse-CHMM emphasizes the diagonal



(a) Emissions of LF BTC

(b) Emissions of LF crunchbase_uncased

Figure 4.10: Emission probabilities of two LFs on CoNLL 2003.

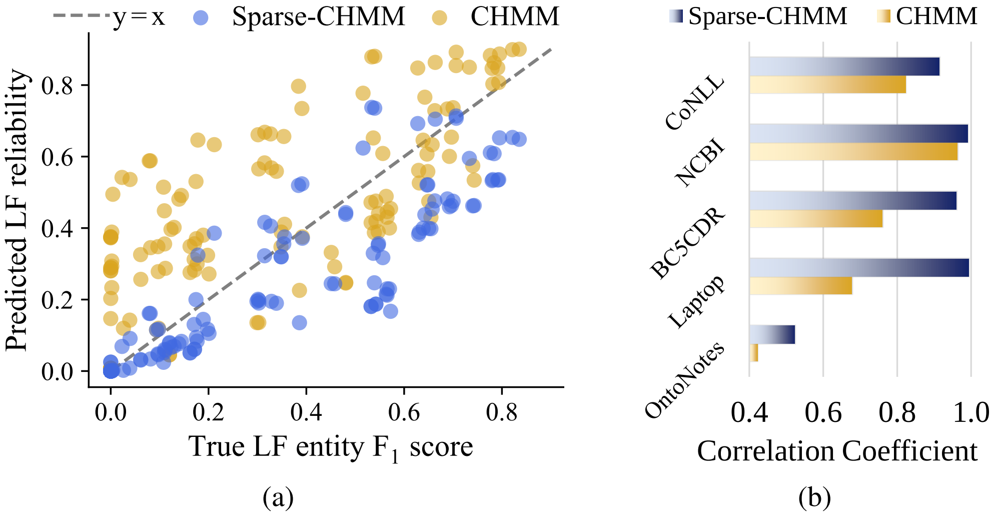


Figure 4.11: Illustration of the correlation between the predicted LF reliability scores and the true LF F_1 scores. Figure a is from the CoNLL 2003 dataset.

and emit-to-O values (the first column). Subsequently, the model refines its emission matrices during stage two by incorporating the addon prior $\boldsymbol{\Delta}$, effectively highlighting prominent off-diagonal values through enhanced Dirichlet parameters. Notably, despite the absence of labeled data, Sparse-CHMM closely aligns its diagonal values with the true emission matrices, demonstrating effective reliability estimation of LFs. These reliability scores not only serve as critical insights into model behavior but also assist in auxiliary tasks, including LF design and evaluation.

Further analysis investigates the correlation between predicted LF reliability and entity F_1 scores of LFs. Figure 4.11a visually depicts this relationship, while Figure 4.11b quantitatively evaluates the correlation strength. The analysis reveals a robust correlation between predicted reliability and actual F_1 scores, achieving correlation coefficients approaching 1.0 across three datasets. Compared to CHMM, Sparse-CHMM shows superior performance in accurately identifying unreliable LFs, a crucial capability for effectively disregarding incorrect observations.

Interestingly, we notice a tendency of Sparse-CHMM to underestimate model reliability scores. Although increasing parameters s and n in Equation 4.3 could theoretically correct this underestimation, such adjustments inadvertently degrade overall model performance.

This suggests that directly aligning emission matrix construction with F_1 scores may require careful scaling or alternative approaches to maintain optimal predictive capabilities.

4.7 Results and Analysis for Autoregressive Zero-Shot Information Extraction

4.7.1 Main Results

Table 4.5 presents the **partial** and **full** matching F_1 scores across different datasets. Overall, G&O-NER consistently outperforms the One-Step approach under both evaluation metrics. Specifically, decoupling task instructions from the organizational prompts leads to an average increase of 15.8% in partial-match F_1 and 12.1% in full-match F_1. Furthermore, the superior performance of G&O compared to the AEiO baseline underscores the critical role of explicitly crafted, entity-specific instructions. A detailed inspection of Figure 4.12 shows that G&O-NER notably improves precision without incurring a significant average reduction in recall, compared to One-Step prompting. This precision enhancement can be attributed to the natural language reasoning capability introduced by CoT, which enables GPT-3.5 to internally verify and refine predictions, resulting in more accurate entity extraction.

4.7.2 Comparison with Weak Supervision

When comparing the results presented in Table 4.5 with those in Table 4.3, a noticeable performance gap emerges between autoregressive LLMs and weakly supervised NER models. Although direct comparisons are inherently challenging—since weak supervision methods rely on carefully designed additional LFs, which are more complex to create compared to simple LLM prompts—it remains evident that autoregressive PLMs have limitations in discriminative tasks with constrained label spaces. Given that LLMs also incur significantly higher inference costs than weakly supervised models, we conclude that G&O-NER should currently be viewed as a complementary strategy rather than a direct substitute for weakly supervised NER models within pre-defined entity recognition tasks. This complementary

Table 4.5: GPT-3.5 F_1 on the NER datasets.

	CoNLL 2003		**BC5CDR**		**NCBI Disease**		**PolyIE**		**Macro Average**	
	Partial	Full	Partial	Full	Partial	Full	Partial	Full	Partial	Full
AEiO	0.5370	0.4965	0.6199	0.5058	-	-	0.1300	0.0935	0.4290	0.3653
One-Step	0.4741	0.4477	0.7030	0.6041	0.6500	0.5131	0.4669	0.3207	0.5735	0.4714
G&O-NER	0.6569	0.6192	0.7610	0.6079	0.6935	0.5047	0.5449	0.3823	0.6641	0.5285
− CoT	0.6572	0.6079	0.6634	0.5544	0.5653	0.4059	0.4551	0.3068	0.5853	0.4688
− clean-up	0.7003	0.6436	0.7421	0.5861	0.6475	0.4541	0.5103	0.3421	0.6501	0.5065
+ CR	0.6775	0.6394	0.7724	0.6186	-	-	0.6011	0.4236	0.6837	0.5605
+ FT	0.7175	0.6800	0.7949	0.6838	0.7703	0.5507	0.7608	0.5533	0.7609	0.6170

relationship suggests that combining these methodologies could potentially leverage their respective strengths effectively.

4.7.3 Ablation Studies

To assess the contribution of specific components within G&O, we conduct two ablation studies focused on the effects of CoT prompting and the clean-up step. Removing the CoT prompts results in an average partial-match F_1 decrease of 11.87%, highlighting the critical role of explicit reasoning in model performance. Similarly, omitting the clean-up component leads to a modest average performance decline of 2.11%. Interestingly, while the individual impact of these components is relatively limited or occasionally negative on general datasets like CoNLL 2003, they yield substantial performance gains on scientific datasets. This discrepancy occurs because LLMs frequently introduce extraneous terms in scientific domains, a problem not typically observed in general entity recognition tasks such as identifying `person` entities. The structured reasoning provided by CoT prompts proves particularly beneficial in scientific contexts, where entities are inherently more complex and nuanced.

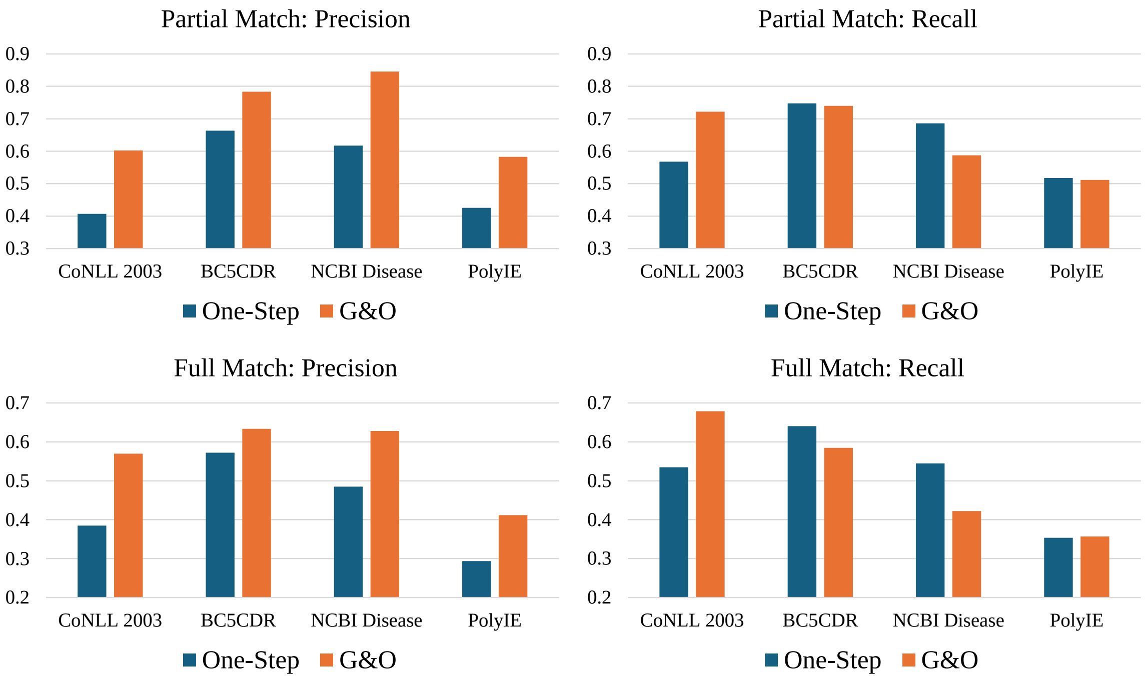


Figure 4.12: Precision and recall on NER datasets.

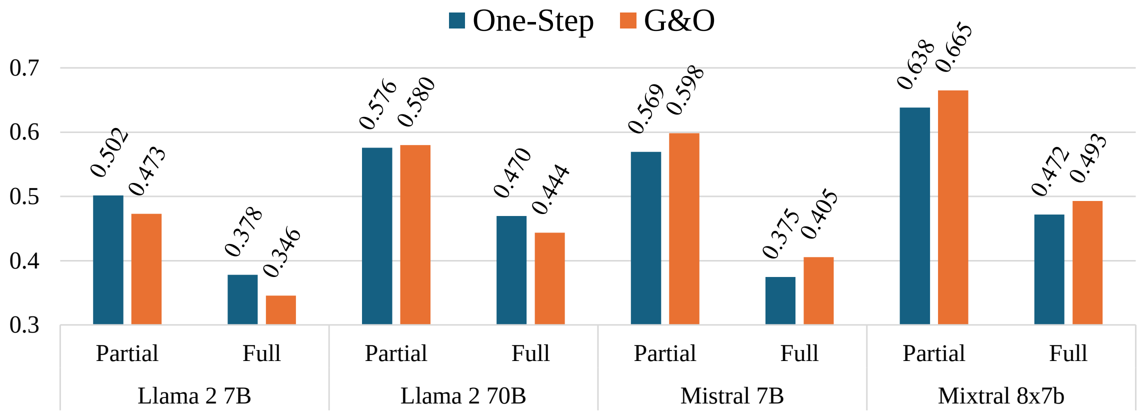


Figure 4.13: F_1 scores of differnt LMs with G&O and One-Step promptings, macro-averaged on the all datasets.

4.7.4 Resolving Entity Type Conflict

Since G&O-NER processes entity types independently, it occasionally results in conflicting labels for a single entity span. To handle these conflicts effectively, we evaluate two distinct strategies:

1. Conflict Resolution (**CR**), which involves explicitly prompting the LLMs to resolve entity-type conflicts as they arise; and

2. BERT Fine-Tuning (**FT**), where we fine-tune a BERT-like encoder [10] using pseudo labels generated by GPT.

As demonstrated in Table 4.5, both approaches significantly enhance the overall performance of G&O-NER, with FT yielding the most pronounced improvements. Beyond resolving type conflicts, FT serves as a robust filtering mechanism that effectively captures stable patterns from GPT-generated pseudo labels, substantially reducing random noise and improving prediction reliability.

4.7.5 Other Language Models

To evaluate the broader applicability of G&O, we examine its performance across 4 open-source LLMs: Llama 2 7B/70B [158], Mistral 7B [19], and Mixtral 8x7B [159], using their respective chat/instruct variants. As illustrated in Figure 4.13, G&O provides relatively modest gains with Llama 2 models, which infrequently generate explicit reasoning. Consequently, in these instances, G&O's behavior closely resembles the simpler One-Step prompting method and suffers more readily from error propagation. In contrast, G&O effectively leverages the advanced reasoning and conversational capabilities of Mi[s/x]tral models, consistently producing clear, structured justifications that lead to notable improvements over One-Step prompting. Thus, G&O is particularly advantageous for LLMs that inherently emphasize multi-step reasoning and conversational coherence.

4.8 Conclusion

In this chapter, we addressed the challenges inherent in weakly supervised NER and explored the promising application of LLMs for structured IE. We introduced CHMM and Sparse-CHMM, two advanced graphical models enhanced by NNs and PLM embeddings, designed to aggregate weak annotations from multiple noisy NER LFs. Specifically, CHMM employs PLM token embeddings and an MLPto estimate token-wise transition and emission probabilities within an HMM framework. Building upon CHMM, Sparse-CHMM further refines the emission probability estimation by leveraging sparse, sentence-level emission matrices derived from LF reliability predictions using PLM embeddings. To robustly handle incorrect entity annotations, Sparse-CHMM incorporates WXOR scores, effectively enhancing off-diagonal emission probabilities. Supported by a carefully structured three-stage training strategy and Dirichlet sampling, Sparse-CHMM consistently surpasses all baseline label models across five diverse datasets. Additionally, the estimated LF reliability scores exhibit a strong correlation with actual LF performance, positioning Sparse-CHMM as a valuable tool for auxiliary tasks, including automated LF generation and systematic evaluation.

For autoregressive zero-shot IE, we introduced G&O, a straightforward yet powerful framework designed to enhance structured prediction capabilities of LLMs. By decomposing the information extraction task into distinct identification and formatting phases, G&O allows LLMs to separately focus on recognizing relevant information and organizing it coherently. Evaluated in zero-shot settings with GPT-3.5, G&O significantly outperforms conventional single-step prompting methods, affirming its effectiveness. Ablation studies reinforce the importance of each stage and underscore the benefit of the two-step methodology, while additional experiments demonstrate G&O's generalizability across multiple LLMs. Performance improvements were further achieved by incorporating effective conflict resolution strategies, such as targeted prompting and fine-tuning, emphasizing G&O's potential as a versatile and scalable solution for structured IE tasks.

In summary, this chapter highlights the potential of combining advanced graphical models with LLMs to tackle the challenges of weakly supervised NER and structured IE. By leveraging the strengths of both approaches, we pave the way for more accurate and efficient information extraction systems that can adapt to a variety of domains and applications.

CHAPTER 5
ENSEMBLES OF LOW-RANK EXPERT ADAPTERS

5.1 Introduction

Building upon insights established in earlier chapters, this chapter explores effective and efficient strategies for fine-tuning LLMs. Although general-domain LLMs such as GPT-4 [160, 17] and Llama [18] exhibit impressive capabilities across various tasks, achieving optimal performance for specialized applications frequently necessitates targeted domain- or task-specific fine-tuning. For example, instruction-following abilities typically require additional tuning on specialized datasets to address nuances and contextual intricacies not fully captured by general-purpose pre-training corpora [22]. Notable datasets used for fine-tuning, such as Alpaca [161], the Pile [162], and Flan [163], highlight the substantial effort dedicated to adapting LLMs to specialized domains and tasks, ranging from medical diagnostics [164] to sophisticated CoT reasoning [20].

Nevertheless, fine-tuning LLMs on extensive, heterogeneous datasets introduces significant challenges, particularly due to the issue of *conflicting gradient directions* [165, 62, 166]. Conflicting gradients arise because different subsets within the training data—even those from a similar domain—can induce model parameter updates in divergent directions, leading to slower convergence and potentially reducing overall model performance. Several recent strategies have been proposed to mitigate this challenge: importance resampling [167] adjusts training distributions to emphasize more beneficial samples, and targeted instruction tuning [62] selectively samples training instances based on gradient feature similarities, isolating more compatible subsets. These approaches demonstrate that carefully selecting subsets of training data can yield superior fine-tuning outcomes compared to training on the entire dataset. However, such methods typically rely heavily on task-specific features or labeled validation data, thus limiting their generalizability and effectiveness when applied to novel tasks.

Rather than fitting a single model to the entire data distribution, MoE frameworks offer an alternative solution by learning multiple specialized modules, often implemented as LoRA experts [23, 168], and dynamically routing inputs to the most relevant expert during inference [169, 170, 171]. Router mechanisms range from learned gating networks [172, 173] to domain-specific heuristics [174, 175], all aiming to optimize computational resources by selectively activating only necessary subsets of experts. Concurrently, Deep Ensemble methods [80, 176] have been explored to improve predictive performance and uncertainty estimation through aggregating multiple models' outputs [177, 178].

Inspired by these developments, we propose ELREA, an innovative ensemble framework explicitly designed to tackle conflicting gradient directions encountered during LLM fine-tuning, without dependence on task-specific assumptions. ELREA begins by training a single *base* LoRA adapter across the entire dataset. Subsequently, it clusters training instances based on the gradient impact they exert on this base module. Each resulting cluster spawns a specialized LoRA expert initialized from the base adapter to preserve computational efficiency. At inference, ELREA dynamically combines the predictions of these specialized experts by computing routing weights that measure the similarity between the test input's gradient profile and the respective cluster profiles. Leveraging insights from Deep Ensemble methods [80, 176], these routing weights can be efficiently precomputed and reused, ensuring negligible computational overhead at runtime. Crucially, ELREA achieves notable data efficiency, as it requires no additional labeling or domain-specific heuristics, making it especially well-suited for dynamic environments where tasks and data distributions frequently change.

In summary, the key contributions and characteristics of ELREA include:

- **Gradient-driven clustering**: ELREA uses gradient profiles from a base LoRA adapter to form specialized expert modules, enabling flexible and efficient adaptation to diverse tasks without requiring additional labels or domain-specific features.

- **Parameter-efficient specialization**: By initializing from a shared base adapter, ELREA minimizes computational overhead while effectively resolving conflicting gradients through dedicated, specialized experts.
- **Efficient inference via ensemble routing**: Lightweight, precomputed routing weights ensure minimal runtime overhead, making ELREA practical for large-scale applications.

Extensive experimental evaluations demonstrate that ELREA consistently surpasses single-adapter baselines and other MoE or self-consistency methods, providing a robust and practical solution for fine-tuning large-scale LLMs on heterogeneous datasets.

5.2 Preliminaries

5.2.1 Language Models and Parameter-Efficient Fine-Tuning

A PLM, denoted as $\mathcal{M}$, learns the language patterns on extensive text corpora $\mathcal{D}_{\text{pre-train}}$ through an unsupervised NTP objective, which minimized the negative NLL of a subsequent token x_t in a length-T sequence $\boldsymbol{x} \in \mathbb{V}^T$ consisting tokens from the vocabulary $\mathbb{V}$ based on the preceding context $\boldsymbol{x}_{<t}$:

$$\mathcal{L}_{\text{NTP}}(\boldsymbol{x}) = -\sum_{t=1}^{T} \log P(x_t|\boldsymbol{x}_{<t}; \boldsymbol{\theta}_{\mathcal{M}}), \tag{5.1}$$

where $\boldsymbol{\theta}_{\mathcal{M}}$ are the network parameters of the LLM. Originally designed for text completion, the pretrained LLMs have been enhanced with instruction-following or task-specific capabilities through targeted fine-tuning [22, 160, 17], expanding their utility across diverse applications. The fine-tuning process frequently adopts the ntp objective, utilizing a smaller, specialized fine-tuning dataset $\mathcal{D}_{\text{ft}}$ that consists of instruction-response pairs $\boldsymbol{x}_{\text{ft}} = (\boldsymbol{x}_{\text{instr}}, \boldsymbol{x}_{\text{resp}})$.

Full-parameter fine-tuning of high-performing Language Models (LMs), which involves calculating $\nabla_{\boldsymbol{\theta}_{\mathcal{M}}}\mathcal{L}_{\text{NTP}}(\boldsymbol{x})$ and updating $\boldsymbol{\theta}_{\mathcal{M}}$ accordingly, is often impractical due to computational constraints arising from their vast number of parameters. To address this issue, Parameter-Efficient Fine-Tuning (PEFT) techniques have been developed [179, 180, 181], with LoRA being a prominent example. LoRA introduces adapter $\boldsymbol{\theta}_{\mathcal{Q}}$ into the LM's linear layers whose weight matrices are, for example, $\boldsymbol{W}_i \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, where i is the layer index and d_{model} is the model dimensionality as defined in [13]. LoRA approximates the weight adjustments during fine-tuning using a low-rank decomposition $\Delta\boldsymbol{W}_i \approx \boldsymbol{A}_i\boldsymbol{B}_i^{\mathsf{T}}$. Here, $\boldsymbol{A}_i, \boldsymbol{B}_i \in \mathbb{R}^{d_{\text{model}} \times r}$ are rank-r adapter matrices with $r \ll d_{\text{model}}$. During fine-tuning, the original weight matrices $\boldsymbol{W}_i$ remain frozen, and only the adapter parameters $\boldsymbol{\theta}_{\mathcal{Q}} \triangleq \bigcup_i \{\boldsymbol{A}_i, \boldsymbol{B}_i\}$ are updated to minimize the NLL loss: $\min_{\boldsymbol{\theta}_{\mathcal{Q}}} \mathcal{L}_{\text{ntp}}(\boldsymbol{x}; \boldsymbol{\theta}_{\mathcal{M}} + \boldsymbol{\theta}_{\mathcal{Q}})$. PEFT significantly reduces the computational demands of fine-tuning by limiting gradient calculations to a smaller set of parameters.

5.2.2 Gradient Feature Calculation and Data Selection

Originally introduced by [182] to estimate the impact of individual training examples on model performance, gradient-based data selection has been further applied to training data selection [183, 62, 184, 185, 171]. Unlike methods based on surface-form textual features—which utilize token statistics or sentence embeddings as selection criteria [126, 167], this approach employs parameter gradients $\nabla_{\boldsymbol{\theta}}$ instead. Specifically, when fine-tuning a LoRA expert $\mathcal{Q}$ using SGD, the gradient feature $\boldsymbol{g}(\boldsymbol{x})$ for each sequence $\boldsymbol{x}$ can be computed as

$$\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{|\boldsymbol{\theta}_{\mathcal{Q}}|} = \operatorname{flatten}\left(\nabla_{\boldsymbol{\theta}_{\mathcal{Q}}}\mathcal{L}_{\text{NTP}}(\boldsymbol{x})\right). \tag{5.2}$$

$\operatorname{flatten}(\cdot)$ denotes the operation that reshapes matrices into vectors and concatenates them. Using this expression, we derive the trajectory influence of a training data point $\boldsymbol{x}_{\text{ft}} \in \mathcal{D}_{\text{ft}}$, quantified by the inner product between its gradient feature and that of a task-specific validation data point $\boldsymbol{x}_{\text{valid}}$. This inner product is then accumulated across training epochs e, each weighted by the average learning rate $\eta^{(e)}$ for that epoch: $\sum_{e=1}^{E} \eta^{(e)} \langle \boldsymbol{g}(\boldsymbol{x}_{\text{ft}}), \boldsymbol{g}(\boldsymbol{x}_{\text{valid}}) \rangle$.

By leveraging this formulation and adapting it to the Adam optimizer (subsection 5.3.2), [62] demonstrate the efficacy of selecting a subset of training data with the highest influence scores for task-specific fine-tuning, achieving performance comparable to that obtained using the complete training dataset.

5.2.3 Mixture of Experts and Ensembles

MoE is an architecture that combines multiple expert models or network modules with a gating network [186, 187]. In the context of LLMs, MoE was first adopted by [170] for instruction-tuning and by [159] for LLM pre-training to reduce inference costs while achieving performance comparable to dense networks. This idea has been further developed in subsequent works [188, 189, 190].

Upon receiving an input, the MoE's gating network routes it to the relevant experts, which could be an entire feed-forward Transformer block [159] or a fine-tuned LoRA expert [191, 192] for LMs. Routing could be either dense or sparse, depending on the fraction of the total experts are activated. The selected experts process the input and provide their outputs, which are aggregated at the end of the layer or block, typically through weighted averaging, to produce the final result. This dynamic and selective activation of experts ensures efficient computation and resource utilization. Mathematically, the output of a mixture of M experts can be expressed as:

$$\mathcal{F}(\boldsymbol{x}) = \sum_{m=1}^{M} \lambda_m(\boldsymbol{x})\mathcal{E}_m(\boldsymbol{x}); \quad \sum_{m=1}^{M} \lambda_m(\boldsymbol{x}) = 1,\ \left|\{m|\lambda_m(\boldsymbol{x}) \neq 0\}_{m=1}^{M}\right| \leq M, \tag{5.3}$$

where $\mathcal{E}_m$ is an expert model, and $0 \leq \lambda_m \leq 1$ is its weight predicted by the gating network. Here we extend the definition of $\boldsymbol{x}$ to any kind of layer input.

On the other hand, Deep Ensembles utilize a collection of multiple models with identical architecture that are trained independently with different parameter initializations [80, 193]. During inference, the last-layer predictions of these models, which could be either pre-activation logits or post-activation probabilities, are averaged to improve the overall performance. Suppose we have N models $\{\mathcal{M}_n\}_{n=1}^N$ in the ensemble, the output would be:

$$\mathcal{M}_{\text{ens}}(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{M}_n(\boldsymbol{x}). \tag{5.4}$$

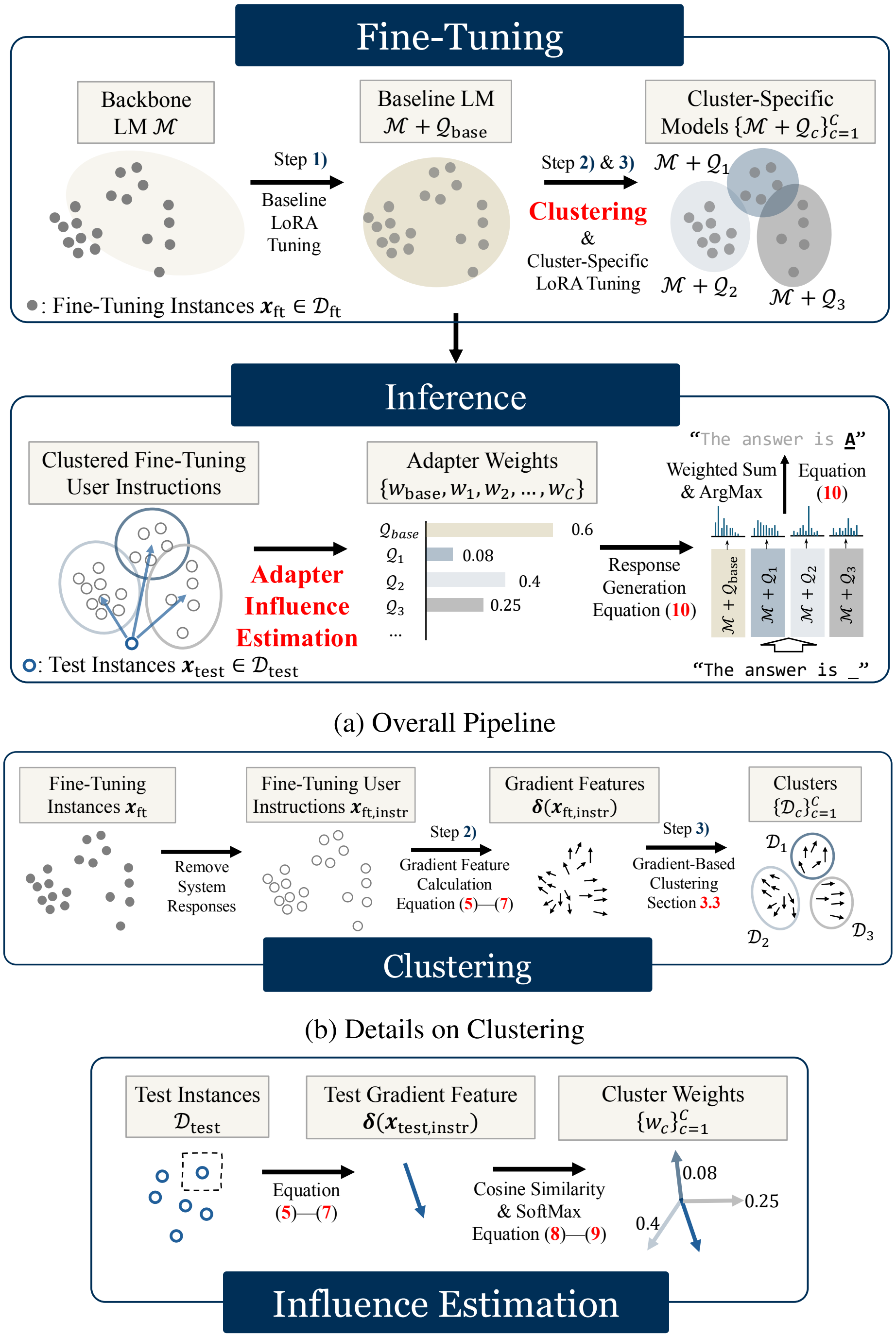
The major differences between MoE and Deep Ensembles are two-fold: 1) MoE uses *trainable* gating networks for model selection, while Deep Ensembles average uniform or predefined weights; 2) MoE conducts output aggregation within layers or blocks, while Deep Ensembles do so at the end of the model. Although MoE can achieve finer-grained routing and potentially superior performance with careful design, Deep Ensembles, as both theoretically and empirically shown, remain the top approach for robustly improving model performance in value prediction and uncertainty estimation, albeit at the cost of reduced efficiency [80, 194, 104, 195, 196, 41].

5.3 Methodology

In this section, we introduce the pipeline of ELREA, designed to enhance the fine-tuning of LLMs for improved downstream tasks by leveraging a mixture of LoRA experts in a Deep Ensembles framework. The pipeline, shown in Figure 5.1, consists of three main steps: 1) full-data adapter tuning, 2) gradient calculation, and 3) clustering and per-cluster fine-tuning. During inference, we estimate the similarity between the gradient of test instructions and the cluster instances to determine the influence of each cluster on the final prediction. The details of each step are elaborated below.

5.3.1 Full-Data Adapter Tuning

The first step involves fine-tuning a base LoRA expert $\mathcal{Q}_{\text{base}}$ from the backbone LM $\mathcal{M}$ on the entire fine-tuning dataset $\mathcal{D}_{\text{ft}}$ for E epochs using the NTP objective (Equation 5.1). This process captures a broader spectrum of general and task-specific knowledge and enhances


Fine-Tuning
Backbone LM $\mathcal{M}$
Step 1)
Baseline LoRA Tuning
Baseline LM $\mathcal{M} + \mathcal{Q}_{\text{base}}$
Step 2) & 3)
Clustering & Cluster-Specific LoRA Tuning
Cluster-Specific Models $\{\mathcal{M} + \mathcal{Q}_c\}_{c=1}^C$
$\mathcal{M} + \mathcal{Q}_1$
$\mathcal{M} + \mathcal{Q}_2$
$\mathcal{M} + \mathcal{Q}_3$
●: Fine-Tuning Instances $\boldsymbol{x}_{\text{ft}} \in \mathcal{D}_{\text{ft}}$
Inference
Clustered Fine-Tuning User Instructions
Adapter Influence Estimation
Adapter Weights $\{w_{\text{base}}, w_1, w_2, \ldots, w_C\}$
$\mathcal{Q}_{base}$ 0.6
$\mathcal{Q}_1$ 0.08
$\mathcal{Q}_2$ 0.4
$\mathcal{Q}_3$ 0.25
...
Response Generation Equation (10)
“The answer is A”
Weighted Sum & ArgMax
Equation (10)
$\mathcal{M} + \mathcal{Q}_{\text{base}}$
$\mathcal{M} + \mathcal{Q}_1$
$\mathcal{M} + \mathcal{Q}_2$
$\mathcal{M} + \mathcal{Q}_3$
“The answer is _”
○: Test Instances $\boldsymbol{x}_{\text{test}} \in \mathcal{D}_{\text{test}}$

(a) Overall Pipeline

Fine-Tuning Instances $\boldsymbol{x}_{\text{ft}}$
Remove System Responses
Fine-Tuning User Instructions $\boldsymbol{x}_{\text{ft,instr}}$
Step 2)
Gradient Feature Calculation Equation (5)—(7)
Gradient Features $\boldsymbol{\delta}(\boldsymbol{x}_{\text{ft,instr}})$
Step 3)
Gradient-Based Clustering Section 3.3
Clusters $\{\mathcal{D}_c\}_{c=1}^C$
$\mathcal{D}_1$
$\mathcal{D}_2$
$\mathcal{D}_3$
Clustering

(b) Details on Clustering

Test Instances $\mathcal{D}_{\text{test}}$
Equation (5)—(7)
Test Gradient Feature $\boldsymbol{\delta}(\boldsymbol{x}_{\text{test,instr}})$
Cosine Similarity & SoftMax Equation (8)—(9)
Cluster Weights $\{w_c\}_{c=1}^C$
0.08
0.25
0.4
Influence Estimation

(c) Details on Influence Estimation

Figure 5.1: The pipeline of ELREA for fine-tuning and inference. The data points (solid and hollow circles) do not necessarily have a geometric correspondence to their gradient directions (arrows).

the model's basic instruction-following abilities. The adapted model checkpoints $\{\mathcal{M} + \mathcal{Q}^{(e)}_{\text{base}}\}_{e=1}^{E}$, where $\mathcal{Q}^{(e)}_{\text{base}}$ denotes the adapter checkpoint at the end of training epoch e, along with the corresponding optimizer states, provide the necessary parameters to calculate the gradient features [62].[1]

5.3.2 Gradient Calculation

With Adam optimizer [197], which is the most adopted for LM fine-tuning, the gradient feature $\boldsymbol{g}(\boldsymbol{x})$ for each sequence $\boldsymbol{x}$ is extended from Equation 5.2 to consider the 1st and 2nd order momentum terms with decay rates β_1 and β_2, as derived by [62]:

$$\begin{aligned} \boldsymbol{g}^{(t)}_{\text{Adam}}(\boldsymbol{x}) &= \eta^{(t)} \cdot \boldsymbol{m}^{(t)} / (\sqrt{\boldsymbol{v}^{(t)}} + \epsilon); \\ \boldsymbol{m}^{(t)} &= (\beta_1 \boldsymbol{m}^{(t-1)} + (1-\beta_1)\boldsymbol{g})/(1-\beta_1^t); \\ \boldsymbol{v}^{(t)} &= (\beta_2 \boldsymbol{v}^{(t-1)} + (1-\beta_2)\boldsymbol{g}^2)/(1-\beta_2^t), \end{aligned} \tag{5.5}$$

where t is the current training step and ϵ is a small constant to prevent division by zero. Each training instance $\boldsymbol{x}_{\text{ft}} \in \mathcal{D}_{\text{ft}}$ is then associated with E gradients $\{\boldsymbol{g}^{(e)}_{\text{Adam}}(\boldsymbol{x}_{\text{ft}})\}_{e=1}^{E}$, each with the dimensionality of the number of total parameters in the adapter $|\boldsymbol{\theta}_{\mathcal{Q}}|$.[2]

Although $|\boldsymbol{\theta}_{\mathcal{Q}}| \ll |\boldsymbol{\theta}_{\mathcal{M}}|$, it is still at a million level scale, which is too large for efficient clustering or similarity computation. Therefore, we follow [62] and apply random projection [198], which is derived from the Johnson-Lindenstrauss lemma [199] stating that sufficiently high-dimensional data points can be projected into lower-dimensional space while approximately preserves pairwise distances between the points, to reduce the dimensionality of the gradient features to $d_{\text{proj}} \ll |\boldsymbol{\theta}_{\mathcal{Q}}|$.

$$\boldsymbol{g}'_{\text{Adam}} = \boldsymbol{R}\boldsymbol{g}_{\text{Adam}}; \quad \boldsymbol{R} \in \{-1, 1\}^{d_{\text{proj}} \times |\boldsymbol{\theta}_{\mathcal{Q}}|}; \; R_{ij} \sim \mathcal{U}(\{-1, 1\}). \tag{5.6}$$

For gradient feature clustering, we first average the gradient features of each instance across all epochs to obtain a single representative feature vector, which is then normalized

[1] Here we extend the definition of the operator "+" between the backbone model and an adapter to denote the addition of the weights of the corresponding network layers [23].

[2] We use the same rank for all adapters, so we do not emphasize the difference of adapters here.

and projected into a $(d_{\text{proj}} - 1)$-dimensional hyper-sphere:

$$\boldsymbol{\delta}(\boldsymbol{x}) = \frac{\boldsymbol{\delta}'(\boldsymbol{x})}{\|\boldsymbol{\delta}'(\boldsymbol{x})\|}; \quad \boldsymbol{\delta}'(\boldsymbol{x}) = \frac{1}{E}\sum_{e=1}^{E} \boldsymbol{g}'^{(e)}_{\text{Adam}}(\boldsymbol{x}) \tag{5.7}$$

as we are only interested in the gradient directions rather than their magnitudes.

ELREA is developed under the assumption that the test distribution is entirely unknown during fine-tuning. Therefore, for both fine-tuning and test instances, we only consider the gradient of the instruction (*i.e.* user-input) tokens $\boldsymbol{x}_{\text{instr}}$ (subsection 5.2.1), excluding the expected system responses even if they are provided in the training data, which is different from [62] who construct the gradients based only on the expected model answers.

5.3.3 Clustering and Per-Cluster Fine-Tuning

We then cluster the training gradient features $\{\boldsymbol{\delta}(\boldsymbol{x}_{\text{ft, instr}}) | \boldsymbol{x}_{\text{ft, instr}} \in \mathcal{D}_{\text{ft}}\}$ into K clusters using the BIRCH algorithm [200]. The BIRCH algorithm is well-suited for large, high-dimensional datasets and demonstrates robustness against outliers. To reduce computational demands, we randomly select 5,000 data points from $\mathcal{D}_{\text{ft}}$ for model fitting. This sample size adequately represents the feature distribution, and we use the resulting model to cluster all gradient features. Preliminary experiments show that the clustering algorithm is robust, *i.e.*, it consistently produces identical or similar clusters when different random seeds are used. As BIRCH does not ensure balanced clusters, we reapply it to clusters exceeding five times the size of the smallest cluster. We iterate this process up to three times, each iteration targeting fewer clusters. Initially targeting 5 clusters, this method typically yields between 8 (after two iterations) and 10 (after three iterations) training clusters $\{\mathcal{D}_c\}_{c=1}^{C}$, where C denotes the final number of clusters.

Within each cluster $\mathcal{D}_c$, we proceed with LoRA fine-tuning from the base checkpoint $\mathcal{Q}^{(E)}_{\text{base}}$, extending for several more epochs at a reduced learning rate utilizing the same NTP objective. This results in a collection of C LoRA experts $\{\mathcal{Q}_c\}_{c=1}^{C}$. Theoretically, each cluster contains training instances with similar gradient directions, which likely exert analogous effects on the model's behavior. Fine-tuning with clustered data aims to direct the model towards a more precise update path, thereby potentially enhancing the model's (*i.e.*, $\mathcal{M} + \mathcal{Q}_c$) performance on specific task types which are unidentified during fine-tuning.

5.3.4 Routing and Inference

To route an input instruction to appropriate expert adapters, we calculate the cosine similarity between the gradient of the instruction $\boldsymbol{\delta}_{\text{test}} \triangleq \boldsymbol{\delta}(\boldsymbol{x}_{\text{test, instr}})$ and the centroid of the gradients within each cluster $\bar{\boldsymbol{\delta}}'_c = \frac{1}{|\mathcal{D}_c|}\sum_{\boldsymbol{x}_i \in \mathcal{D}_c} \boldsymbol{\delta}(\boldsymbol{x}_{i,\text{instr}})$. The normalized form of $\bar{\boldsymbol{\delta}}_c$ is given by:

$$\bar{\boldsymbol{\delta}}_c = \frac{\bar{\boldsymbol{\delta}}'_c}{\|\bar{\boldsymbol{\delta}}'_c\|}; \quad \bar{\boldsymbol{\delta}}'_c = \frac{1}{|\mathcal{D}_c|}\sum_{\boldsymbol{x}_i \in \mathcal{D}_c} \boldsymbol{\delta}(\boldsymbol{x}_{i,\text{instr}}). \tag{5.8}$$

Here, the cosine similarity simply becomes the inner product of these two normalized vectors: $\cos(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c) = \langle \boldsymbol{\delta}_{\text{t}}, \bar{\boldsymbol{\delta}}_c \rangle$. When the projection dimensionality d_{proj} is high, the similarity may suffer from the curse of dimensionality, where the gaps between the similarities to different cluster centroids may become too small. To address this issue, we standardize the cosine similarities across clusters before employing a SoftMax function on the standardized similarities $\cos'(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c)$ across clusters to determine their respective weights:

$$w_c = \frac{\exp(\cos'(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c))}{\sum_{c'=1}^{C} \exp(\cos'(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_{c'}))}; \quad \cos'(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c) = \frac{\cos(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c) - \mu_{\text{test}}}{\sigma_{\text{test}}}, \tag{5.9}$$

where μ_{test} and σ_{test} are the mean and standard deviation of the cosine similarities across clusters.

Besides the cluster-specific adapters $\{\mathcal{Q}_c\}_{c=1}^{C}$, we also incorporate the base adapter $\mathcal{Q}_{\text{base}}$ during inference to leverage the general knowledge captured from the entire dataset. This is particularly crucial when the test instruction diverges significantly from all training instances, indicated by $\max_c\{\cos(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c)\} < \tau$, where τ is some threshold. We quantify the influence of the base adapter as $w_{\text{base}} = 1 - \max_c\{\cos(\boldsymbol{\delta}_{\text{test}}, \bar{\boldsymbol{\delta}}_c)\}$. Therefore, we assemble $C+1$ adapters during inference, with the final prediction for the next token being the ArgMax of the weighted sum of output logits from each adapter:

$$\hat{x}_t = \arg\max_{x_t} \left(w_{\text{base}}(\mathcal{M} + \mathcal{Q}_{\text{base}})(x_t|\boldsymbol{x}_{<t}) + \sum_{c=1}^{C} w_c(\mathcal{M} + \mathcal{Q}_c)(x_t|\boldsymbol{x}_{<t}) \right), \tag{5.10}$$

which is a combination of Equation 5.3 and Equation 5.4. x_t is categorical, while $\mathcal{M}(x_t|\boldsymbol{x}_{<t})$ denotes the output pre-activation logit of categorical token x_t given the context tokens $\boldsymbol{x}_{<t}$ from the language model $\mathcal{M}$. In Equation 5.10 we get $\hat{x}_t$, we append it to the context tokens $\boldsymbol{x}_{<t+1} = (\boldsymbol{x}_{<t}, \hat{x}_t)$ for *all* adapters in the ensemble and repeat the process until the end of the sequence is reached. As we are not dealing with probabilities here, the weights do not need to sum to 1, *i.e.* $w_{\text{base}} + \sum_{c=1}^{C} w_c \neq 1$.

Unlike the LoRA MoE approaches (subsection 5.2.3), which utilizes a gating network for layer-wise routing with predictions aggregated post-layer, ELREA resembles Deep Ensembles in its routing and aggregation strategy but uses LoRA adapters as ensemble components, and hence the name.

5.4 Experiment Setup

5.4.1 Datasets

We conduct experiments on two categories of tasks: 1) general language understanding and reasoning, and 2) mathematical reasoning. Following [62], we use Flan V2 [163], CoT [20], Dolly-15k [201], and OpenAssistant Conversations [202] for fine-tuning, and evaluate on MMLU [203] and BIG-bench Hard (BBH; [127, 128]). These training and test sets have no distribution overlap, enabling fair assessments of model generalization.

For mathematical reasoning, we unify GSM8k [50], MathQA [205], SVAMP [204], and MATH [51] into the MATH-Combined dataset, formatted similarly to MATH. Since MATH-Combined contains in-domain test points, it enables insights into how specialized data selection affects fine-tuning. Please refer to section D.1 for dataset details and Ta-

Table 5.1: Dataset statistics. The fine-tuning datasets are mixed together and randomly shuffled before being used for model fine-tuning or clustering.

	Dataset	**Source**	**# Instance**	l_{instr}(a)	l_{resp}(a)
General Language Understanding and Reasoning					
Fine-Tune	Dolly-15k	[201]	15,011	72.41	60.12
	OpenAssistant	[202]	55,668	20.14	113.09
	CoT	[20]	100,000	168.70	34.94
	Flan V2	[163]	100,000	216.59	16.71
Test	BBH	[128]	6,511	64.87(b)	105.51
	MMLU	[203]	14,042	88.53(b)	1
Mathematical Reasoning (MATH-Combined)					
Fine-Tune & Test	MATH	[51]	7,500 & 1,000	32.69	88.47
	GSM8k	[50]	7,441 & 1,000	45.19	56.93
	SVAMP	[204]	677 & 280	31.66	28.15
	MathQA	[205]	26,287 & 998	38.39	69.09

(a) These numbers represent the average number of words (character strings separated by whitespace and newline characters) in the instruction and response sequences. They are generally smaller than the number of tokens.

(b) These numbers do not include the in-context examples; if the examples are considered, the counts will be approximately $3\times$ larger for BBH and $5\times$ larger for MMLU.

ble 5.1 for statistics.

5.4.2 Model and Fine-Tuning

We fine-tune the `gemma-1.1-2b-it` model [206] with rank-8 LoRA adapters on all linear layers, modifying about 0.39% of the model parameters. For each task category, we train the base adapter, $\mathcal{Q}_{\text{base}}$, for 2 epochs using Adam with an initial learning rate of 5×10^{-5}, decaying linearly to zero. Cluster-wise adapters, $\mathcal{Q}_c$, are initialized from $\mathcal{Q}_{\text{base}}$ and fine-tuned for the same duration at a slightly lower learning rate of 2×10^{-5}. No task-specific validation data are used. The maximum sequence length during training is 2,048 tokens, with a batch size of 16 sequences. Following [62], we set the gradient projection dimensionality d_{proj} to 8,192 for best performance. Additional details are in section D.2 and section D.3.

5.4.3 Inference and Evaluation

For general reasoning tasks (BBH, MMLU), we employ up to three in-context examples from BBH and five from MMLU, while mathematical reasoning (MATH-Combined) is zero-shot. We limit instruction lengths to 1,200 tokens and responses to 848 for MATH-Combined and BBH, and use 1,800 and 248 tokens for MMLU. If the total length exceeds these limits, we reduce in-context examples. We decode greedily (temperature = 0) with the largest feasible batch size. Accuracy is computed via parsing code from [51] for MATH-Combined, and regular expressions for MMLU and BBH. Because our approach differs from [206], final results may not match theirs.

5.4.4 Baselines

Our main baseline, $\mathcal{M} + \mathcal{Q}_{\text{base}}$, is trained on the entire dataset. $\mathcal{M} + \mathcal{Q}_{\text{dataset}}$ uses adapters fine-tuned separately on each MATH-Combined subset. We also evaluate the backbone $\mathcal{M}$ without fine-tuning. For MoE, we compare MoE Routing (layer-level routing with the same weights as ELREA), MoE Merging through parameter averaging, and Mixture of LoRA Experts (MoLE) [192], which adopts layer-wise gating. From ensembling methods, Self-Consistency [132] runs five inference passes at temperature 1, while LoRA Ensembles [177] trains three additional adapters and averages predictions. The Instruction Embedding baseline replaces instruction gradients with pretrained sentence embeddings for clustering and routing.

We also include two ablations: Random Cluster preserves cluster sizes but randomly assigns training samples (approximating Deep Ensembles under equal training cost), and Uniform Weights assigns equal inference weights to all clusters. See section D.4 for further details.

Table 5.2: Comparison of test set accuracies (in %) across various MATH-Combined subsets, along with the **micro**-average. Gray rows indicate the primary baseline; blue rows highlight ELREA.

LoRA Rank	Methods	MATH	GSM8k	SVAMP	MathQA	Average[a]
			Gemma-2b			
$r = 8$	$\mathcal{M} + \mathcal{Q}_{\text{base}}$	9.2	22.1	46.07	16.83	18.61
	$\mathcal{M} + \mathcal{Q}_{\text{dataset}}$	7.3	25.7	45.00	16.73	19.01 (+ 0.40)
	MoE Routing	9.2	22.7	48.21	16.23	18.79 (+ 0.18)
	MoE Merging	9.1	23.1	48.21	15.73	18.73 (+ 0.12)
	MoLE	8.8	21.6	46.43	15.53	17.99 (– 0.62)
	LoRA Ensembles	9.3	24.7	47.50	16.73	19.55 (+ 0.94)
	Self-Consistency	5.9	14.3	44.64	10.32	13.12 (– 5.49)
	Instruction Embedding	9.8	24.1	46.79	16.83	19.46 (+ 0.85)
	ELREA	9.1	25.9	49.64	18.04	**20.41 (+ 1.80)**
	Random Cluster	9.1	25.1	48.21	18.84	20.30 (+ 1.69)
	Uniform Weights	9.6	25.2	47.50	18.04	20.16 (+ 1.55)
$r = 64$	$\mathcal{M} + \mathcal{Q}_{\text{base}}$	10.8	32.7	55.36	27.56	26.39
	$\mathcal{M} + \mathcal{Q}_{\text{dataset}}$	10.8	33.0	52.14	27.66	26.24 (– 0.15)
	MoE Routing	11.7	31.9	60.36	26.95	26.66 (+ 0.27)
	MoE Merging	11.4	32.0	60.36	26.85	26.57 (+ 0.18)
	MoLE	10.7	31.7	56.07	25.35	25.49 (– 0.90)
	LoRA Ensembles	12.1	31.8	60.00	28.06	27.06 (+ 0.67)
	Self-Consistency	9.3	28.5	60.36	21.84	23.34 (– 3.05)
	Instruction Embedding	11.2	31.7	60.71	28.46	26.94 (+ 0.55)
	ELREA	12.5	32.6	57.86	28.36	**27.33 (+ 0.94)**
	Random Cluster	11.5	32.8	59.64	27.05	26.87 (+ 0.48)
	Uniform Weights	11.4	31.5	60.00	27.15	26.48 (+ 0.24)
			Gemma2-9b			
$r = 8$	$\mathcal{M} + \mathcal{Q}_{\text{base}}$	37.9	78.7	84.64	50.30	58.11
	ELREA	37.4	78.6	86.43	52.00	58.60 (+ 0.49)
$r = 64$	$\mathcal{M} + \mathcal{Q}_{\text{base}}$	37.4	81.3	86.07	57.82	61.17
	ELREA	36.8	80.7	87.50	59.32	61.38 (+ 0.21)

[a] The number in parentheses indicates the improvement over the corresponding baseline $\mathcal{M} + \mathcal{Q}_{\text{base}}$.

Table 5.3: Comparison of test set exact-match accuracy (in %) on BBH and MMLU, and the **macro**-averaged result. We also include the backbone $\mathcal{M}$ for reference.

LoRA Rank	**Methods**	**BBH**	**MMLU**	**Macro Average**
N/A	Backbone $\mathcal{M}^{(a)}$	9.17	9.12	9.15
$r = 8$	$\mathcal{M} + \mathcal{Q}_{\text{base}}$	27.20	33.73	30.47
	MoE Routing	27.46 (+ 0.26)	34.21 (+ 0.48)	30.84 (+ 0.37)
	MoE Merging	27.13 (– 0.07)	33.98 (+ 0.25)	30.36 (+ 0.09)
	MoLE	26.40 (– 0.80)	34.19 (+ 0.46)	30.30 (– 0.17)
	Self-Consistency	23.74 (– 3.46)	32.88 (– 0.85)	28.31 (– 2.16)
	Instruction Embedding	26.50 (– 0.70)	34.76 (+ 1.03)	30.63 (+ 0.16)
	ELREA	**28.03 (+ 0.83)**	**34.84 (+ 1.11)**	**31.44 (+ 0.97)**
	Random Cluster	27.72 (+ 0.52)	34.56 (+ 0.83)	31.14 (+ 0.67)
	Uniform Weights	27.32 (+ 0.12)	34.33 (+ 0.60)	30.83 (+ 0.36)

[(a)] A large portion of responses are unparsable, leading to an accuracy lower than random guess.

5.5 Results and Discussion

5.5.1 Main Results

Table 5.2 summarizes the test set accuracy across different subsets of MATH-Combined and presents micro-averaged results. ELREA consistently demonstrates superior performance over baseline methods on most subsets, with only occasional, minor exceptions. On average, ELREA achieves notable improvements of 9.67% and 3.56% over $\mathcal{M} + \mathcal{Q}_{\text{base}}$ at ranks $r = 8$ and $r = 64$, respectively, without requiring additional training data or external knowledge sources.

Further evaluations shown in Table 5.3 reinforce ELREA's robustness across general language understanding and reasoning tasks, even when test conditions differ significantly from those encountered during fine-tuning, echoing findings reported in [62]. An interesting observation emerges when comparing $\mathcal{M} + \mathcal{Q}_{\text{dataset}}$ and $\mathcal{M} + \mathcal{Q}_{\text{base}}$: the former does not consistently outperform the latter. This suggests that adopting a generalized knowledge-extraction approach (illustrated in Figure D.1) may occasionally surpass dataset-specific expert models, highlighting the value of broader knowledge integration.

Despite using precomputed routing weights, both the MoE Routing and Merging frameworks still show improvements over baseline models. This enhancement likely stems from the ensemble benefits provided by combining multiple expert modules. Conversely, the MoLE baseline, which incorporates a trainable router, consistently underperforms relative to $\mathcal{M} + \mathcal{Q}_{\text{base}}$. We hypothesize that the complexity introduced by multiple LoRA experts applied across numerous linear layers (as detailed in section D.2) creates a highly intricate routing landscape. This complexity may hinder effective optimization, resulting in suboptimal solutions characterized by training data overfitting and poor generalization. Addressing this challenge may necessitate more sophisticated network architectures or refined training methodologies.

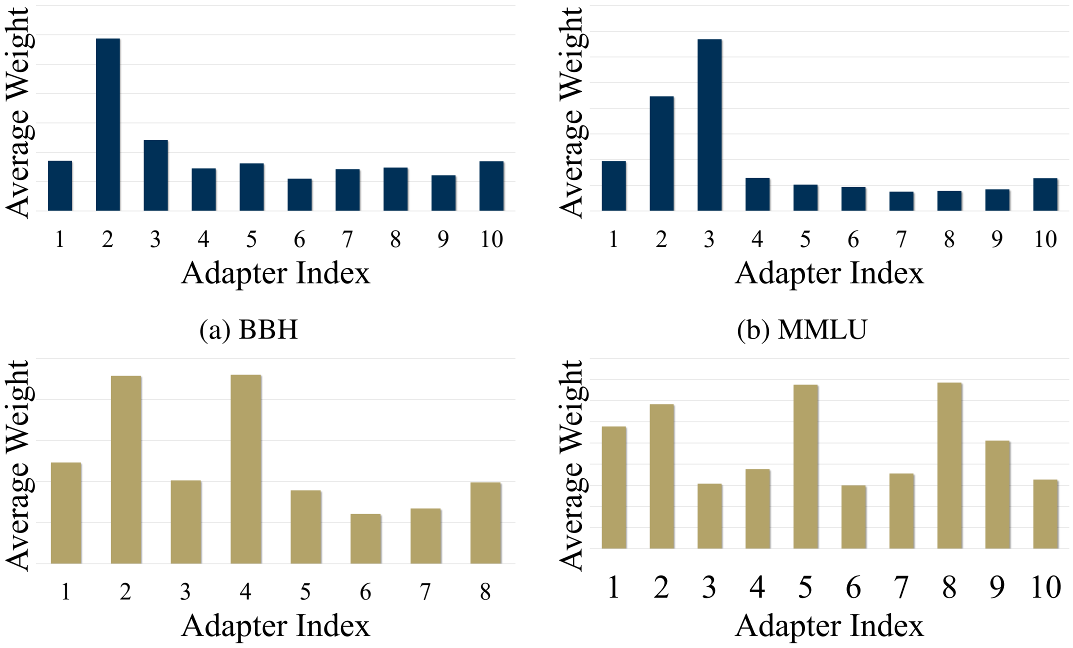
On the other hand, the classical LoRA Ensembles approach demonstrates considerable robustness, consistently outperforming $\mathcal{M} + \mathcal{Q}_{\text{base}}$ despite higher computational costs. These findings align with discussions in subsection 5.2.3, underscoring the strength and effectiveness of ELREA's ensemble strategy. In contrast, the Self-Consistency method produces relatively poorer results due to significant variability across runs, particularly at elevated sampling temperatures. Additionally, the Instruction Embedding baseline trails ELREA, emphasizing the importance of precise gradient profiling to effectively extract and leverage specialized expertise.

When evaluating the Gemma2-9b architecture, ELREA continues to outperform the base adapter, albeit with a narrower performance margin. The inherently advanced capabilities of Gemma2-9b for capturing task-specific knowledge without extensive fine-tuning diminish the incremental benefits provided by ELREA [130]. This observation suggests that ELREA's advantages are more pronounced when the base model is less optimized for the task at hand or when dealing with datasets characterized by greater complexity and diversity.

Table 5.4: The performance of ELREA with different clustering methods. The results use Gemma-2b backbone, LoRA rank $r = 64$, and number of clusters $C = 10$.

Methods	MATH	GSM8k	SVAMP	MathQA	Average
ELREA	12.5	32.6	57.86	28.36	27.33
BIRCH w/ 256-d PCA	10.3	32.1	60.00	26.95	26.27
K-means[(a)]	10.7	32.9	58.93	28.46	27.00
K-means w/o grad norm (Equation 5.7)[(a)]	10.8	32.3	58.21	27.56	26.51

[(a)] Both use 256-d Principal Component Analysis (PCA) for dimensionality reduction. Otherwise the gradient outliers result in multiple clusters with few data points.



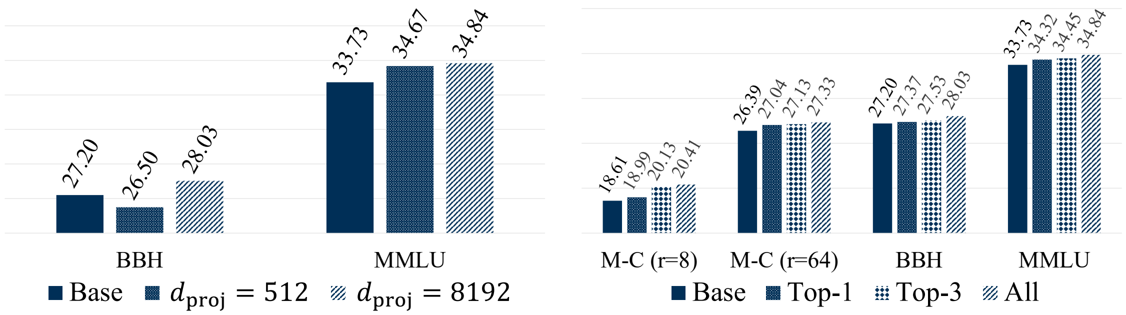
(a) BBH

(b) MMLU

(c) M-C ($r = 8$)

(d) M-C ($r = 64$)

Figure 5.2: Average weight distribution across clusters for different datasets and LoRA ranks. Only relative values matter. "M-C" represents MATH-Combined.



(a) Gradient projection dimensionality

(b) Number of top-k experts

Figure 5.3: Effects of gradient projection dimensionality and selection of top-k experts during inference on model performance.

5.5.2 Ablation Studies

An examination of Table 5.2 and Table 5.3 reveals that the gradient-based clustering method consistently achieves superior performance compared to random clustering. This emphasizes the effectiveness of gradient-based clustering in accurately isolating in-domain, task-specific subsets of data for fine-tuning. Nevertheless, the performance advantage of ELREA over the Random Cluster baseline is occasionally modest. This outcome is expected since Random Cluster essentially emulates a Deep Ensembles approach, a robust baseline that leverages extensive and diverse training data effectively.

Furthermore, the relatively poor performance observed in the Uniform Weights baseline underscores the critical role of a carefully engineered routing mechanism within ELREA. Figure 5.2 illustrates the average cluster weight distribution, highlighting distinct patterns across datasets. Specifically, the MATH-Combined test set exhibits a more balanced activation across experts, whereas BBH and MMLU display pronounced skewness toward one or two dominant clusters. Such skewed distributions typically occur when test examples closely align with only a small portion of the training data, possibly dominated by specific clusters. This may partially explain why the LESS method, as in [62], can outperform baseline approaches by strategically leveraging smaller but more relevant training subsets.

As noted by [62], the dimensionality of gradient projections (d_{proj}) significantly impacts the effectiveness of similarity-based matching between training and test instances. Figure 5.3a confirms this finding within the context of ELREA. When reducing d_{proj} from 8,192 to 512, a notable decline in exact-match accuracy occurs, likely due to the loss of critical, fine-grained gradient information resulting from the random projection. Interestingly, on the BBH dataset, this dimensionality reduction results in ELREA performing worse than both the base adapter and the Random Cluster baseline. Moreover, Table 5.4 demonstrates that employing alternative dimensionality reduction methods such as PCA or using k-means clustering instead of gradient-based clustering similarly reduces performance. These findings collectively highlight the importance of maintaining representative gradient features for effective clustering and subsequent routing decisions; any simplifications or inappropriate dimensionality reductions significantly compromise model accuracy.

Additionally, Figure 5.3b illustrates how increasing the number of top-k experts selected during inference generally enhances ELREA's performance. This improvement suggests that the model benefits from aggregating insights from a more diverse group of experts, even when individual contributions may be minor. Nonetheless, selecting fewer experts provides efficiency gains at inference time. Thus, carefully balancing the trade-off between inference speed and performance becomes essential in practical deployments.

5.6 Conclusion

In this chapter we introduced ELREA, a framework designed to address the impact of conflicting gradient directions during the fine-tuning of LLMs across diverse datasets for data-efficient applications. ELREA develops multiple LoRA experts, each optimized for a specific data cluster with similar gradient profiles. These adapters collaboratively generate predictions by dynamically adjusting their contributions based on the input's gradient characteristics, effectively resolving gradient conflicts without the need for task-specific data features or validation sets. With only one single training session, ELREA enhances the adaptability of models to new or evolving tasks and outperforms traditional LoRA adapters and other ensemble techniques across a variety of applications. Ablation studies confirm that both the ensemble structure and the gradient-based clustering and routing mechanisms are integral to ELREA's effectiveness. These findings underscore the framework's potential for efficient and scalable application of LLMs in practical settings.

CHAPTER 6
CONCLUSION

6.1 Summary

In this thesis, we investigated two fundamental directions towards building reliable and data-efficient PLMs for a wide array of downstream tasks: We first focused on quantifying confidence in PLMs, particularly in high-stakes domains such as healthcare, finance, and materials science, where overconfident or miscalibrated models can lead to severe consequences. In Chapter chapter 2, we introduced MUBen, a systematic benchmarking framework for UQ in molecular property prediction. MUBen evaluates and compares diverse UQ methods across multiple SOTA pre-trained molecular representation models, thereby distilling best practices for building reliable, uncertainty-aware molecular models. Extending beyond discriminative settings, Chapter chapter 3 proposed UQAC, a novel pipeline for confidence estimation in autoregressive LLMs. UQAC leverages an "attention chain" mechanism to approximate the intractable marginalization over all possible reasoning paths, resulting in more robust and interpretable final-answer confidence scores.

We then turned our attention to methods that enable effective model adaptation when labeled data are scarce or expensive to acquire. In Chapter chapter 4, we tackled the zero-shot NER problem, developing weak supervision methods—CHMM and Sparse-CHMM—to integrate multiple weak LFs without requiring any manually labeled sequences. We also explored G&O, which leverages autoregressive LLMs for zero-shot IE tasks. Finally, in Chapter chapter 5, we introduced ELREA, a fine-tuning strategy that clusters training examples by gradient directions into distinct "expertise regions." By training separate expert adapters and automatically routing new inputs to the most appropriate adapter at inference time, ELREA scales efficiently and mitigates the negative effects of conflicting gradient updates.

Collectively, these contributions form a minimally supervised and uncertainty-aware toolkit for PLMs, addressing both the challenge of unreliable model outputs and the high cost of obtaining large-scale labeled data. The thesis thereby aims to promote safer, more transparent, and more accessible deployment of PLMs in real-world scenarios.

6.2 Future Works

While our research provides strong groundwork, several promising directions remain open for future exploration:

Extending Uncertainty Quantification to New Settings Although our work has primarily examined UQ in discriminative molecular tasks and autoregressive LLMs, many PLMs operate in domains where data modalities go beyond textual or molecular inputs (*e.g.*, multimodal biomedical records, sensor-based time-series data). Developing scalable UQ techniques that incorporate diverse data modalities and complex causal relationships could lead to more robust decision-making in areas such as healthcare, autonomous driving, or materials discovery. Additionally, exploring tighter integration between UQ methods and instruction-tuned LLMs may further improve calibration and interpretability of large-scale generative models.

Interactive and Human-in-the-Loop Approaches In addition, many real-world applications demand iterative and interactive analysis rather than one-shot predictions. It is worth exploring a way of adopting the well-performed UQ systems to the data-efficient training procedure, and how our frameworks can incorporate domain experts' feedback to refine model calibration and adapt to shifting data distributions on the fly. Incorporating human insights on model reliability might strengthen the trustworthiness of PLMs and lead to faster discovery cycles in scientific domains.

In summary, we envision that the methodological contributions in this thesis—covering UQ benchmarks, confidence estimation strategies for autoregressive LLMs, weak supervision approaches, and data-efficient fine-tuning—can serve as a robust foundation for building more reliable, adaptable, and transparent PLMs. By addressing the challenges of miscalibrated predictions and label scarcity, we hope to pave the way for a new generation of AI-driven solutions with broader utility and safer real-world deployment, ultimately contributing to the continued integration of PLMs into high-stakes applications.

REFERENCES

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013.

[2] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds., ACL, 2014, pp. 1532–1543.

[3] A. Frome *et al.*, “Devise: A deep visual-semantic embedding model,” in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2121–2129.

[4] D. Kiela and L. Bottou, “Learning image embeddings using convolutional neural networks for improved multi-modal semantics,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds., ACL, 2014, pp. 36–45.

[5] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 2787–2795.

[6] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, Eds., AAAI Press, 2015, pp. 2181–2187.

[7] W. Hu *et al.*, “Strategies for pre-training graph neural networks,” in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[8] D. Koge, N. Ono, M. Huang, M. Altaf-Ul-Amin, and S. Kanaya, “Embedding of molecular structure using molecular hypergraph variational autoencoder with metric learning,” *Molecular informatics*, vol. 40, no. 2, p. 2 000 203, 2021.

[9] M. E. Peters *et al.*, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds., Association for Computational Linguistics, 2018, pp. 2227–2237.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *ACL 2019*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[11] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, “Improving language understanding by generative pre-training,” 2018.

[12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon *et al.*, Eds., 2017, pp. 5998–6008.

[14] J. Kaplan *et al.*, “Scaling laws for neural language models,” *CoRR*, vol. abs/2001.08361, 2020. arXiv: 2001.08361.

[15] L. Gao, J. Schulman, and J. Hilton, “Scaling laws for reward model overoptimization,” in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, 2023, pp. 10 835–10 866.

[16] T. B. Brown *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[17] OpenAI, “GPT-4 technical report,” *CoRR*, vol. abs/2303.08774, 2023. arXiv: 2303.08774.

[18] H. Touvron *et al.*, “Llama: Open and efficient foundation language models,” *CoRR*, vol. abs/2302.13971, 2023. arXiv: 2302.13971.

[19] A. Q. Jiang *et al.*, “Mistral 7b,” *CoRR*, vol. abs/2310.06825, 2023. arXiv: 2310.06825.

[20] J. Wei *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *NeurIPS*, 2022.

[21] N. Shazeer *et al.*, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[22] L. Ouyang *et al.*, “Training language models to follow instructions with human feedback,” in *NeurIPS 2022*, 2022.

[23] E. J. Hu *et al.*, “Lora: Low-rank adaptation of large language models,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[24] Z. Shao *et al.*, “Deepseekmath: Pushing the limits of mathematical reasoning in open language models,” *CoRR*, vol. abs/2402.03300, 2024. arXiv: 2402.03300.

[25] DeepSeek-AI, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” *CoRR*, vol. abs/2501.12948, 2025. arXiv: 2501.12948.

[26] S. Chithrananda, G. Grand, and B. Ramsundar, “Chemberta: Large-scale self-supervised pretraining for molecular property prediction,” *CoRR*, vol. abs/2010.09885, 2020. arXiv: 2010.09885.

[27] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, “SMILES-BERT: large scale unsupervised pre-training for molecular property prediction,” in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, BCB 2019, Niagara Falls, NY, USA, September 7-10, 2019*, X. M. Shi, M. Buck, J. Ma, and P. Veltri, Eds., ACM, 2019, pp. 429–436.

[28] S. Liu, H. Wang, W. Liu, J. Lasenby, H. Guo, and J. Tang, “Pre-training molecular graph representation with 3d geometry,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[29] G. Zhou *et al.*, “Uni-mol: A universal 3d molecular representation learning framework,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[30] J. Jumper *et al.*, “Highly accurate protein structure prediction with alphafold,” *nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[31] J. Abramson *et al.*, “Accurate structure prediction of biomolecular interactions with alphafold 3,” *Nature*, vol. 630, no. 8016, pp. 493–500, 2024.

[32] B. Chen *et al.*, "Xtrimopglm: Unified 100b-scale pre-trained transformer for deciphering the language of protein," *CoRR*, vol. abs/2401.06199, 2024. arXiv: 2401.06199.

[33] Y. Liu, S. Ding, S. Zhou, W. Fan, and Q. Tan, "Moleculargpt: Open large language model (LLM) for few-shot molecular property prediction," *CoRR*, vol. abs/2406.12950, 2024. arXiv: 2406.12950.

[34] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988. eprint: https://doi.org/10.1021/ci00057a005.

[35] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar, "Chemberta-2: Towards chemical foundation models," *CoRR*, vol. abs/2209.01712, 2022. arXiv: 2209.01712.

[36] Y. Rong *et al.*, "Self-supervised graph transformer on large-scale molecular data," in *NeurIPS 2020*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[37] H. Wang *et al.*, "Efficient evolutionary search over chemical space with large language models," *CoRR*, vol. abs/2406.16976, 2024. arXiv: 2406.16976.

[38] K. Valmeekam, M. Marquez, and S. Kambhampati, "Can large language models really improve by self-critiquing their own plans?" *CoRR*, vol. abs/2310.08118, 2023. arXiv: 2310.08118.

[39] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1321–1330.

[40] L. Kong, H. Jiang, Y. Zhuang, J. Lyu, T. Zhao, and C. Zhang, "Calibrated language model fine-tuning for in- and out-of-distribution data," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Association for Computational Linguistics, 2020, pp. 1326–1340.

[41] Y. Li *et al.*, "MUBen: Benchmarking the uncertainty of molecular representation models," *Transactions on Machine Learning Research*, 2024.

[42] Z. C. Lipton, "The mythos of model interpretability," *Commun. ACM*, vol. 61, no. 10, pp. 36–43, 2018.

[43] N. S. Eyke, W. H. Green, and K. F. Jensen, "Iterative experimental design based on active machine learning reduces the experimental burden associated with reaction screening," *React. Chem. Eng.*, vol. 5, pp. 1963–1972, 10 2020.

[44] A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia, and C. W. Coley, "Evidential deep learning for guided molecular property prediction and discovery," *ACS Central Science*, vol. 7, no. 8, pp. 1356–1367, 2021. eprint: https://doi.org/10.1021/acscentsci.1c00546.

[45] X. Shen, Z. Chen, M. Backes, and Y. Zhang, "In chatgpt we trust? measuring and characterizing the reliability of chatgpt," *CoRR*, vol. abs/2304.08979, 2023. arXiv: 2304.08979.

[46] Y. Li, H. Wang, and C. Zhang, "Assessing logical puzzle solving in large language models: Insights from a minesweeper case study," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, K. Duh, H. Gómez-Adorno, and S. Bethard, Eds., Association for Computational Linguistics, 2024, pp. 59–81.

[47] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *ACL 2011*, Portland, Oregon, USA, Jun. 2011, pp. 541–550.

[48] B. Boecking, W. Neiswanger, E. P. Xing, and A. Dubrawski, "Interactive weak supervision: Learning useful heuristics for data labeling," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

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

[50] K. Cobbe *et al.*, "Training verifiers to solve math word problems," *CoRR*, vol. abs/2110.14168, 2021. arXiv: 2110.14168.

[51] D. Hendrycks *et al.*, "Measuring mathematical problem solving with the MATH dataset," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, J. Vanschoren and S. Yeung, Eds., 2021.

[52] J. J. Cheung *et al.*, "Polyie: A dataset of information extraction from polymer material scientific literature," *CoRR*, vol. abs/2311.07715, 2023. arXiv: 2311.07715.

[53] A. Toland *et al.*, "Accelerated scheme to predict ring-opening polymerization enthalpy: Simulation-experimental data fusion and multitask machine learning," *The Journal of Physical Chemistry A*, vol. 127, no. 50, pp. 10 709–10 716, 2023.

[54] Y. Li, R. Qiang, L. Moukheiber, and C. Zhang, *Language model uncertainty quantification with attention chain*, 2025. arXiv: 2503.19168 `[cs.CL]`.

[55] A. Ratner, C. D. Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, Barcelona, Spain: Curran Associates Inc., 2016, pp. 3574–3582, ISBN: 9781510838819.

[56] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," *Proc. VLDB Endow.*, vol. 11, no. 3, pp. 269–282, Nov. 2017.

[57] P. Lison, J. Barnes, A. Hubin, and S. Touileb, "Named entity recognition without labelled data: A weak supervision approach," in *ACL 2020*, Online: Association for Computational Linguistics, Jul. 2020, pp. 1518–1533.

[58] P. Lison, J. Barnes, and A. Hubin, "Skweak: Weak supervision made easy for NLP," in *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, H. Ji, J. C. Park, and R. Xia, Eds., Association for Computational Linguistics, 2021, pp. 337–346.

[59] 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 *ACL/IJCNLP 2021*, 2021, pp. 6178–6190.

[60] Y. Li, L. Song, and C. Zhang, "Sparse conditional hidden markov model for weakly supervised named entity recognition," in *KDD '22*, A. Zhang and H. Rangwala, Eds., ACM, 2022, pp. 978–988.

[61] Y. Li, R. Ramprasad, and C. Zhang, "A simple but effective approach to improve structured language model output for information extraction," *CoRR*, vol. abs/2402.13364, 2024. arXiv: 2402.13364.

[62] M. Xia, S. Malladi, S. Gururangan, S. Arora, and D. Chen, "LESS: selecting influential data for targeted instruction tuning," *CoRR*, vol. abs/2402.04333, 2024. arXiv: 2402.04333.

[63] Y. Li, V. R. Gao, C. Zhang, and M. Torkamani, "Ensembles of low-rank expert adapters," in *The Thirteenth International Conference on Learning Representations*, 2025.

[64] W. P. Walters and R. Barzilay, "Applications of deep learning in molecule generation and molecular property prediction," *Accounts of Chemical Research*, vol. 54, no. 2, pp. 263–270, 2021, PMID: 33370107. eprint: https://doi.org/10.1021/acs.accounts.0c00699.

[65] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[66] H. Stärk *et al.*, "3d infomax improves gnns for molecular property prediction," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., ser. Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 20 479–20 502.

[67] Z. Wu *et al.*, "Moleculenet: A benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018.

[68] P. Li *et al.*, "An effective self-supervised framework for learning expressive molecular global representations to drug discovery," *Briefings in Bioinformatics*, vol. 22, no. 6, bbab109, May 2021, bbab109. eprint: https://academic.oup.com/bib/article-pdf/22/6/bbab109/41087439/si_bbab109.pdf.

[69] K. Schütt, P. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon *et al.*, Eds., 2017, pp. 991–1001.

[70] Y. Liu, L. Wang, M. Liu, X. Zhang, B. Oztekin, and S. Ji, "Spherical message passing for 3d graph networks," *CoRR*, vol. abs/2102.05013, 2021. arXiv: 2102.05013.

[71] J. Gasteiger, F. Becker, and S. Günnemann, "Gemnet: Universal directional graph neural networks for molecules," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 6790–6802.

[72] N. Thomas *et al.*, "Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds," *CoRR*, vol. abs/1802.08219, 2018. arXiv: 1802.08219.

[73] Y. Liao and T. E. Smidt, "Equiformer: Equivariant graph attention transformer for 3d atomistic graphs," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[74] W. Du *et al.*, "A new perspective on building efficient and expressive 3d equivariant graph neural networks," *CoRR*, vol. abs/2304.04757, 2023. arXiv: 2304.04757.

[75] J. Noh, G. H. Gu, S. Kim, and Y. Jung, "Uncertainty-quantified hybrid machine learning/density functional theory high throughput screening method for crystals," *Journal of Chemical Information and Modeling*, vol. 60, no. 4, pp. 1996–2003, 2020.

[76] D. E. Graff, E. I. Shakhnovich, and C. W. Coley, "Accelerating high-throughput virtual screening through molecular pool-based active learning," *Chem. Sci.*, vol. 12, pp. 7866–7881, 22 2021.

[77] D. van Tilborg, A. Alenicheva, and F. Grisoni, "Exposing the limitations of molecular machine learning with activity cliffs," *Journal of Chemical Information and Modeling*, vol. 62, no. 23, pp. 5938–5951, 2022, PMID: 36456532. eprint: https://doi.org/10.1021/acs.jcim.2c01073.

[78] J. Gawlikowski *et al.*, "A survey of uncertainty in deep neural networks," *Artif. Intell. Rev.*, vol. 56, no. Suppl 1, pp. 1513–1589, Jul. 2023.

[79] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML 2016*, M. Balcan and K. Q. Weinberger, Eds., ser. JMLR Workshop and Conference Proceedings, vol. 48, JMLR.org, 2016, pp. 1050–1059.

[80] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *NeurIPS 2017*, I. Guyon *et al.*, Eds., 2017, pp. 6402–6413.

[81] M. Sensoy, L. M. Kaplan, and M. Kandemir, "Evidential deep learning to quantify classification uncertainty," in *NeurIPS 2018*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 3183–3193.

[82] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[83] J. Busk, P. B. Jørgensen, A. Bhowmik, M. N. Schmidt, O. Winther, and T. Vegge, "Calibrated uncertainty for molecular property prediction using ensembles of message passing neural networks," *Mach. Learn. Sci. Technol.*, vol. 3, no. 1, p. 15 012, 2022.

[84] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1263–1272.

[85] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[86] A. Deshwal, C. M. Simon, and J. R. Doppa, "Bayesian optimization of nanoporous materials," *Molecular Systems Design & Engineering*, vol. 6, no. 12, pp. 1066–1086, 2021.

[87] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, Mar. 2007.

[88] J. Quiñonero-Candela, C. E. Rasmussen, F. Sinz, O. Bousquet, and B. Schölkopf, "Evaluating predictive uncertainty challenge," in *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, Springer Berlin Heidelberg, 2006, pp. 1–27.

[89] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, Jan. 1950.

[90] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, Eds., vol. 29, AAAI Press, Feb. 2015, pp. 2901–2907.

[91] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," in *ICML 2018*, J. G. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 2801–2809.

[92] P. Thölke and G. D. Fabritiis, "Equivariant transformers for neural network based molecular potentials," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[93] S. Zaidi *et al.*, “Pre-training via denoising for molecular property prediction,” in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[94] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[95] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, pp. 2999–3007.

[96] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, “Calibrating deep neural networks using focal loss,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[97] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural networks,” *CoRR*, vol. abs/1505.05424, 2015. arXiv: 1505.05424.

[98] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *ICLR 2014*, Y. Bengio and Y. LeCun, Eds., 2014.

[99] M. Welling and Y. W. Teh, “Bayesian learning via stochastic gradient langevin dynamics,” in *ICML 2011*, L. Getoor and T. Scheffer, Eds., Omnipress, 2011, pp. 681–688.

[100] A. C. Damianou and N. D. Lawrence, “Deep gaussian processes,” in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, ser. JMLR Workshop and Conference Proceedings, vol. 31, JMLR.org, 2013, pp. 207–215.

[101] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, “A simple baseline for bayesian uncertainty in deep learning,” in *NeurIPS 2019*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 13 132–13 143.

[102] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, “Averaging weights leads to wider optima and better generalization,” in *UAI 2018*, A. Globerson and R. Silva, Eds., AUAI Press, 2018, pp. 876–885.

[103] T. G. Dietterich, “Ensemble methods in machine learning,” in *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23,*

2000, Proceedings, J. Kittler and F. Roli, Eds., ser. Lecture Notes in Computer Science, vol. 1857, Springer, 2000, pp. 1–15.

[104] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *CoRR*, vol. abs/1912.02757, 2019. arXiv: 1912.02757.

[105] X. Fang *et al.*, "Geometry-enhanced molecular representation learning for property prediction," *Nat. Mach. Intell.*, vol. 4, no. 2, pp. 127–134, 2022.

[106] R. C. Fox, "The evolution of medical uncertainty," *The Milbank Memorial Fund Quarterly. Health and Society*, pp. 1–49, 1980.

[107] A. Simpkin and R. Schwartzstein, "Tolerating uncertainty—the next medical revolution?" *New England Journal of Medicine*, vol. 375, no. 18, 2016.

[108] J. Li, X. Cheng, X. Zhao, J. Nie, and J. Wen, "Halueval: A large-scale hallucination evaluation benchmark for large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Association for Computational Linguistics, 2023, pp. 6449–6464.

[109] H. Xu *et al.*, "Rejection improves reliability: Training LLMs to refuse unknown questions using RL from knowledge feedback," in *First Conference on Language Modeling*, 2024.

[110] L. Kuhn, Y. Gal, and S. Farquhar, "Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[111] S. Kadavath *et al.*, "Language models (mostly) know what they know," *CoRR*, vol. abs/2207.05221, 2022. arXiv: 2207.05221.

[112] A. Malinin and M. J. F. Gales, "Uncertainty estimation in autoregressive structured prediction," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[113] B. Hou, Y. Liu, K. Qian, J. Andreas, S. Chang, and Y. Zhang, "Decomposing uncertainty for large language models through input clarification ensembling," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, OpenReview.net, 2024.

[114] M. Jiang *et al.*, "Calibrating language models via augmented prompt ensembles," 2023.

[115] C. Ling *et al.*, “Uncertainty quantification for in-context learning of large language models,” in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, K. Duh, H. Gómez-Adorno, and S. Bethard, Eds., Association for Computational Linguistics, 2024, pp. 3357–3370.

[116] Z. Lin, S. Trivedi, and J. Sun, “Generating with confidence: Uncertainty quantification for black-box large language models,” *Trans. Mach. Learn. Res.*, vol. 2024, 2024.

[117] J. Chen and J. Mueller, “Quantifying uncertainty in answers from any language model and enhancing their trustworthiness,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, L. Ku, A. Martins, and V. Srikumar, Eds., Association for Computational Linguistics, 2024, pp. 5186–5200.

[118] J. Duan *et al.*, “Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, L. Ku, A. Martins, and V. Srikumar, Eds., Association for Computational Linguistics, 2024, pp. 5050–5063.

[119] Z. Lin, S. Trivedi, and J. Sun, “Contextualized sequence likelihood: Enhanced confidence scores for natural language generation,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, Y. Al-Onaizan, M. Bansal, and Y. Chen, Eds., Association for Computational Linguistics, 2024, pp. 10 351–10 368.

[120] S. Lin, J. Hilton, and O. Evans, “Teaching models to express their uncertainty in words,” *Trans. Mach. Learn. Res.*, vol. 2022, 2022.

[121] K. Tian *et al.*, “Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Association for Computational Linguistics, 2023, pp. 5433–5442.

[122] M. Xiong *et al.*, “Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms,” in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, OpenReview.net, 2024.

[123] A. Amayuelas, K. Wong, L. Pan, W. Chen, and W. Y. Wang, "Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models," in *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, L. Ku, A. Martins, and V. Srikumar, Eds., Association for Computational Linguistics, 2024, pp. 6416–6432.

[124] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[125] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds., The Association for Computational Linguistics, 2015, pp. 1412–1421.

[126] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Association for Computational Linguistics, 2019, pp. 3980–3990.

[127] B.-b. authors, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *Transactions on Machine Learning Research*, 2023.

[128] M. Suzgun *et al.*, "Challenging big-bench tasks and whether chain-of-thought can solve them," in *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds., Association for Computational Linguistics, 2023, pp. 13 003–13 051.

[129] Llama Team, "The llama 3 herd of models," *CoRR*, vol. abs/2407.21783, 2024. arXiv: 2407.21783.

[130] Gemma Team, "Gemma 2: Improving open language models at a practical size," *CoRR*, vol. abs/2408.00118, 2024. arXiv: 2408.00118.

[131] Qwen Team, "Qwen2.5 technical report," *CoRR*, vol. abs/2412.15115, 2024. arXiv: 2412.15115.

[132] X. Wang *et al.*, "Self-consistency improves chain of thought reasoning in language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[133] J. Li *et al.*, “Biocreative V CDR task corpus: A resource for chemical disease relation extraction,” *Database J. Biol. Databases Curation*, vol. 2016, 2016.

[134] R. I. Dogan, R. Leaman, and Z. Lu, “NCBI disease corpus: A resource for disease name recognition and concept normalization,” *J. Biomed. Informatics*, vol. 47, pp. 1–10, 2014.

[135] R. Weischedel *et al.*, “Ontonotes release 5.0 ldc2013t19,” Philadelphia: Linguistic Data Consortium, 2013.

[136] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, “SemEval-2014 task 4: Aspect based sentiment analysis,” in *SemEval 2014*, Dublin, Ireland, Aug. 2014, pp. 27–35.

[137] E. Kim, K. Huang, A. Saunders, A. McCallum, G. Ceder, and E. Olivetti, “Materials synthesis insights from scientific literature via text extraction and machine learning,” *Chemistry of Materials*, pp. 9436–9444, 2017.

[138] A. T. Nguyen, B. Wallace, J. J. Li, A. Nenkova, and M. Lease, “Aggregating and predicting sequence labels from crowd annotations,” in *ACL 2017*, Vancouver, Canada, Jul. 2017, pp. 299–309.

[139] E. Safranchik, S. Luo, and S. H. Bach, “Weakly supervised sequence tagging from noisy rules,” in *AAAI 2020*, 2020, pp. 5570–5578.

[140] J. Parker and S. Yu, “Named entity recognition through deep representation learning and weak supervision,” in *Findings of ACL-IJCNLP 2021*, Online, Aug. 2021, pp. 3828–3839.

[141] O. Lan, X. Huang, B. Y. Lin, H. Jiang, L. Liu, and X. Ren, “Learning to contextually aggregate multi-source supervision for sequence labeling,” in *ACL 2020*, Online, Jul. 2020, pp. 2134–2146.

[142] S. Wang *et al.*, “GPT-NER: named entity recognition via large language models,” *CoRR*, vol. abs/2304.10428, 2023. arXiv: 2304.10428.

[143] R. Han, T. Peng, C. Yang, B. Wang, L. Liu, and X. Wan, “Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors,” *CoRR*, vol. abs/2305.14450, 2023. arXiv: 2305.14450.

[144] T. Xie, Q. Li, J. Zhang, Y. Zhang, Z. Liu, and H. Wang, “Empirical study of zero-shot NER with chatgpt,” in *EMNLP 2023*, Association for Computational Linguistics, 2023, pp. 7935–7956.

[145] K. Zhang, B. J. Gutierrez, and Y. Su, “Aligning instruction tasks unlocks large language models as zero-shot relation extractors,” in *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds., Association for Computational Linguistics, 2023, pp. 794–812.

[146] X. Wang *et al.*, “Instructuie: Multi-task instruction tuning for unified information extraction,” *CoRR*, vol. abs/2304.08085, 2023. arXiv: 2304.08085.

[147] W. Zhou, S. Zhang, Y. Gu, M. Chen, and H. Poon, “Universalner: Targeted distillation from large language models for open named entity recognition,” *CoRR*, vol. abs/2308.03279, 2023. arXiv: 2308.03279.

[148] O. Sainz, I. García-Ferrero, R. Agerri, O. L. de Lacalle, G. Rigau, and E. Agirre, “Gollie: Annotation guidelines improve zero-shot information-extraction,” *CoRR*, vol. abs/2310.03668, 2023. arXiv: 2310.03668.

[149] U. Zaratiana, N. Tomeh, P. Holat, and T. Charnois, “Gliner: Generalist model for named entity recognition using bidirectional transformer,” *CoRR*, vol. abs/2311.08526, 2023. arXiv: 2311.08526.

[150] Y. Jiao *et al.*, “Instruct and extract: Instruction tuning for on-demand information extraction,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Association for Computational Linguistics, 2023, pp. 10 030–10 051.

[151] M. Jankowiak and F. Obermeyer, “Pathwise derivatives beyond the reparameterization trick,” in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 2240–2249.

[152] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” in *NeurIPS 2022*, 2022.

[153] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, “An explanation of in-context learning as implicit bayesian inference,” in *ICLR 2022*, OpenReview.net, 2022.

[154] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition,” in *HLT-NAACL 2003*, 2003, pp. 142–147.

[155] R. I. Dogan, R. Leaman, and Z. Lu, "NCBI disease corpus: A resource for disease name recognition and concept normalization," *J. Biomed. Informatics*, vol. 47, pp. 1–10, 2014.

[156] J. Lee *et al.*, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Sep. 2019. eprint: https://academic.oup.com/bioinformatics/article-pdf/36/4/1234/32527770/btz682.pdf.

[157] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3615–3620.

[158] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *CoRR*, vol. abs/2307.09288, 2023. arXiv: 2307.09288.

[159] A. Q. Jiang *et al.*, "Mixtral of experts," *CoRR*, vol. abs/2401.04088, 2024. arXiv: 2401.04088.

[160] OpenAI. "Introducing ChatGPT." (Accessed on Jun 18, 2023). (2022), (visited on 06/18/2023).

[161] R. Taori *et al.*, *Stanford alpaca: An instruction-following llama model*, https://github.com/tatsu-lab/stanford_alpaca, 2023.

[162] L. Gao *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *CoRR*, vol. abs/2101.00027, 2021. arXiv: 2101.00027.

[163] S. Longpre *et al.*, "The flan collection: Designing data and methods for effective instruction tuning," in *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, 2023, pp. 22 631–22 648.

[164] K. Singhal *et al.*, "Towards expert-level medical question answering with large language models," *CoRR*, vol. abs/2305.09617, 2023. arXiv: 2305.09617.

[165] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[166] S. Chen, Z. Jie, and L. Ma, "Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms," *CoRR*, vol. abs/2401.16160, 2024. arXiv: 2401.16160.

[167] S. M. Xie, S. Santurkar, T. Ma, and P. Liang, "Data selection for language models via importance resampling," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[168] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[169] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, 120:1–120:39, 2022.

[170] S. Shen *et al.*, "Flan-moe: Scaling instruction-finetuned language models with sparse mixture of experts," *CoRR*, vol. abs/2305.14705, 2023. arXiv: 2305.14705.

[171] L. Yang *et al.*, "Solving token gradient conflict in mixture-of-experts for large vision-language model," *CoRR*, vol. abs/2406.19905, 2024. arXiv: 2406.19905.

[172] T. Luo *et al.*, "Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models," *CoRR*, vol. abs/2402.12851, 2024. arXiv: 2402.12851.

[173] D. Li *et al.*, "Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts," *CoRR*, vol. abs/2404.15159, 2024. arXiv: 2404.15159.

[174] M. Muqeeth, H. Liu, Y. Liu, and C. Raffel, "Learning to route among specialized experts for zero-shot generalization," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, OpenReview.net, 2024.

[175] Y. Liu *et al.*, "Intuition-aware mixture-of-rank-1-experts for parameter efficient finetuning," *CoRR*, vol. abs/2404.08985, 2024. arXiv: 2404.08985.

[176] M. Havasi *et al.*, "Training independent subnetworks for robust prediction," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[177] X. Wang, L. Aitchison, and M. Rudolph, “Lora ensembles for large language model fine-tuning,” *CoRR*, vol. abs/2310.00035, 2023. arXiv: 2310.00035.

[178] J. Lu, Z. Pang, M. Xiao, Y. Zhu, R. Xia, and J. Zhang, “Merge, ensemble, and cooperate! A survey on collaborative strategies in the era of large language models,” *CoRR*, vol. abs/2407.06089, 2024. arXiv: 2407.06089.

[179] N. Houlsby *et al.*, “Parameter-efficient transfer learning for NLP,” in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 2790–2799.

[180] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Association for Computational Linguistics, 2021, pp. 4582–4597.

[181] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, “Towards a unified view of parameter-efficient transfer learning,” in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022.

[182] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, “Estimating training data influence by tracing gradient descent,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.

[183] Y. Gou *et al.*, “Mixture of cluster-conditional lora experts for vision-language instruction tuning,” *CoRR*, vol. abs/2312.12379, 2023. arXiv: 2312.12379.

[184] X. Pan, L. Huang, L. Kang, Z. Liu, Y. Lu, and S. Cheng, “G-DIG: towards gradient-based diverse and high-quality instruction data selection for machine translation,” *CoRR*, vol. abs/2405.12915, 2024. arXiv: 2405.12915.

[185] Z. Liu, R. Ke, F. Jiang, and H. Li, “Take the essence and discard the dross: A rethinking on data selection for fine-tuning large language models,” *CoRR*, vol. abs/2406.14115, 2024. arXiv: 2406.14115.

[186] P. T. Szymanski and M. D. Lemmon, “Adaptive mixtures of local experts are source coding solutions,” in *Proceedings of International Conference on Neural Networks*

(ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993, IEEE, 1993, pp. 1391–1396.

[187] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, 1994.

[188] T. Zhu *et al.*, "Llama-moe: Building mixture-of-experts from llama with continual pre-training," *CoRR*, vol. abs/2406.16554, 2024. arXiv: 2406.16554.

[189] D. Dai *et al.*, "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models," *CoRR*, vol. abs/2401.06066, 2024. arXiv: 2401.06066.

[190] F. Xue *et al.*, "Openmoe: An early effort on open mixture-of-experts language models," *CoRR*, vol. abs/2402.01739, 2024. arXiv: 2402.01739.

[191] S. Dou *et al.*, "Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment," *CoRR*, vol. abs/2312.09979, 2023. arXiv: 2312.09979.

[192] X. Wu, S. Huang, and F. Wei, "Mixture of lora experts," *CoRR*, vol. abs/2404.13628, 2024. arXiv: 2404.13628.

[193] A. Gleave and G. Irving, "Uncertainty estimation for language reward models," *CoRR*, vol. abs/2203.07472, 2022. arXiv: 2203.07472.

[194] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 8803–8812.

[195] K. Fang, Q. Tao, X. Huang, and J. Yang, "Revisiting deep ensemble for out-of-distribution detection: A loss landscape perspective," *CoRR*, vol. abs/2310.14227, 2023. arXiv: 2310.14227.

[196] S. Pitis, M. R. Zhang, A. Wang, and J. Ba, "Boosted prompt ensembles for large language models," *CoRR*, vol. abs/2304.05970, 2023. arXiv: 2304.05970.

[197] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[198] P. Kanerva, J. Kristoferson, and A. Holst, “Random indexing of text samples for latent semantic analysis,” in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 22, 2000.

[199] W. B. Johnson and J. Lindenstrauss, “Extensions of lipschitz mappings into hilbert space,” *Contemporary mathematics*, vol. 26, pp. 189–206, 1984.

[200] T. Zhang, R. Ramakrishnan, and M. Livny, “Birch: An efficient data clustering method for very large databases,” in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’96, Montreal, Quebec, Canada: Association for Computing Machinery, 1996, pp. 103–114, ISBN: 0897917944.

[201] M. Conover *et al.* “Free dolly: Introducing the world’s first truly open instruction-tuned llm.” (2023), (visited on 06/30/2023).

[202] A. Köpf *et al.*, “Openassistant conversations - democratizing large language model alignment,” in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[203] D. Hendrycks *et al.*, “Measuring massive multitask language understanding,” in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[204] A. Patel, S. Bhattamishra, and N. Goyal, “Are NLP models really able to solve simple math word problems?” 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. 2080–2094.

[205] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, 2019, pp. 2357–2367.

[206] Gemma Team, “Gemma: Open models based on gemini research and technology,” *CoRR*, vol. abs/2403.08295, 2024. arXiv: 2403.08295.

[207] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32: Annual Con-*

ference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[208] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[209] Y. Liu *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. arXiv: 1907.11692.

[210] Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang, and S. H. Bryant, "Pubchem: A public information system for analyzing bioactivities of small molecules," *Nucleic Acids Res.*, vol. 37, no. Web-Server-Issue, pp. 623–633, 2009.

[211] T. Wolf *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *CoRR*, vol. abs/1910.03771, 2019. arXiv: 1910.03771.

[212] T. Sterling and J. J. Irwin, "Zinc 15 - ligand discovery for everyone," *Journal of Chemical Information and Modeling*, vol. 55, no. 11, pp. 2324–2337, 2015, PMID: 26479676. eprint: https://doi.org/10.1021/acs.jcim.5b00559.

[213] A. Gaulton *et al.*, "Chembl: A large-scale bioactivity database for drug discovery," *Nucleic acids research*, vol. 40, no. D1, pp. D1100–D1107, 2012.

[214] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, 2016. arXiv: 1606.08415 [cs.LG].

[215] H. L. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service.," *Journal of Chemical Documentation*, vol. 5, no. 2, pp. 107–113, 1965. eprint: https://doi.org/10.1021/c160017a018.

[216] K. Yang *et al.*, "Correction to analyzing learned molecular representations for property prediction," *Journal of Chemical Information and Modeling*, vol. 59, no. 12, pp. 5304–5305, 2019, PMID: 31814400. eprint: https://doi.org/10.1021/acs.jcim.9b01076.

[217] M. Nakata and T. Shimazaki, "Pubchemqc project: A large-scale first-principles electronic structure database for data-driven chemistry," *J. Chem. Inf. Model.*, vol. 57, no. 6, pp. 1300–1308, 2017.

[218] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "OGB-LSC: A large-scale challenge for machine learning on graphs," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS*

Datasets and Benchmarks 2021, December 2021, virtual, J. Vanschoren and S. Yeung, Eds., 2021.

[219] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[220] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[221] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[222] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2575–2583.

[223] J. Harrison, J. Willes, and J. Snoek, "Variational bayesian last layers," in *The Twelfth International Conference on Learning Representations*, 2023.

[224] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[225] Y. Wen, D. Tran, and J. Ba, "Batchensemble: An alternative approach to efficient ensemble and lifelong learning," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[226] Z. Li, K. Ren, Y. Yang, X. Jiang, Y. Yang, and D. Li, "Towards inference efficient deep ensemble learning," in *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, B. Williams, Y. Chen, and J. Neville, Eds., AAAI Press, 2023, pp. 8711–8719.

[227] Y. Wang *et al.*, "How far can camels go? exploring the state of instruction tuning on open resources," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New*

Orleans, LA, USA, December 10 - 16, 2023, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[228] Anthropic, *Introducing the next generation of claude*, 2024.

VITA

Yinghao Li earned his Bachelor of Science from the School of Instrument Science and Engineering at Southeast University in Nanjing, and his Master of Science from the School of Electrical and Computer Engineering at Georgia Institute of Technology in Atlanta. During his master's program, he specialized in Language Generation under the guidance of Dr. Chao Zhang. He is currently pursuing his Ph.D. in Machine Learning at Georgia Institute of Technology, working or has been working with Dr. Chao Zhang, Dr. Rampi Ramprasad, and Prof. Le Song. His research primarily focuses on Text Generation, Weakly Supervised Named Entity Recognition, Uncertainty Estimation, and the understanding, alignment, and application of Large Language Models.

Appendices

Appendix A

Benchmarking Molecular Representation Uncertainties

A.1 Dataset Details

In this section, we provide details about the datasets used in our study. We follow the approach of previous works [105, 29] and select a subset of widely used and publicly available datasets from the MoleculeNet benchmark [67]. The datasets cover both classification and regression tasks from 4 property categories, including Physiology, Biophysics, Physical Chemistry and Quantum Mechanics. The classification datasets include:

- **BACE** provides binary binding properties for a set of inhibitors of human β-secretase 1 (BACE-1) from experimental values from the published papers;
- **BBBP (Blood-Brain Barrier Penetration)** studies the classification of molecules by their permeability of the blood-brain barrier;
- **ClinTox** consists of two classification tasks for drugs: whether they are absent of clinical toxicity and whether they are approved by the FDA;
- **Tox21 (Toxicology in the 21st Century)** consists of qualitative toxicity measurements of 8,014 compounds on 12 different targets;

- **ToxCast** provides toxicology data of 8,576 compounds on 617 different targets;
- **SIDER (Side Effect Resource)** contains marketed drugs and adverse drug reactions (ADR) extracted from package inserts;
- **HIV**, introduced by the Drug Therapeutics Program (DTP) AIDS Antiviral Screen, contains compounds that are either active or inactive against HIV replication;
- **MUV (Maximum Unbiased Validation)**, a challenging benchmark dataset selected from PubChem BioAssay for validation of virtual screening techniques, contains 93,087 compounds tested for activity against 17 different targets.

The regression datasets include:

- **ESOL** provides experimental values for water solubility data for 1,128 compounds;
- **FreeSolv (Free Solvation Database)** contains experimental and calculated hydration free energies for 643 small molecules;
- **Lipophilicity** contains experimental results of octanol/water distribution coefficient for 4,200 compounds;
- **QM7/8/9** are subsets of GDB-13 and GDB-17 databases containing quantum mechanical calculations of energies, enthalpies, and dipole moments for organic molecules with up to 7/8/9 "heavy" atoms, respectively. For QM9, we follow previous works and select 3 tasks that predict properties (homo, lumo, and gap) with a similar quantitative range out of the total 12 [105, 29].

A summary of the dataset statistics is provided in Table 2.3. It is worth noting that some datasets, such as HIV and MUV, exhibit a high degree of class imbalance. This characteristic adds further challenges to the tasks of molecular property prediction and uncertainty quantification.

Scaffold Splitting MUBen primarily utilizes a dataset split based on molecule scaffolds, effectively segregating training, validation, and test sets to maximize feature separation. This approach generates OOD test sets, crucial for assessing the model's ability to handle patterns not encountered during the fine-tuning process. We adhere to the standard 8:1:1 ratio for training, validation, and test splits across all datasets. The raw datasets can be accessed on the MoleculeNet website.[1] Additionally, we employ the pre-processed versions provided by [29], using the identical dataset splits outlined in their study.[2]

Random Splitting To investigate the impact of dataset splitting methods, we conducted experiments on 4 classification datasets (BACE, BBBP, ClinTox, Tox21) and 4 regression datasets (ESOL, FreeSolv, Lipophilicity, QM7) using random splitting. Despite their relatively small sizes, these datasets provide a reliable representation of model performance. Each dataset was divided into three separate training-validation-test sets with an 8:1:1 ratio, utilizing random seeds of 0, 1, and 2. We trained each backbone-UQ combination once per split and averaged the results across the three runs to establish the final metrics for each dataset. Deep Ensembles was the only exception; it underwent three training cycles per dataset split, using different seeds, resulting in a total of 9 training cycles per dataset. All other training configurations remained consistent with those used in the scaffold-split experiments.

Binning Test Data by Similarity to Training Scaffolds Our experimental analysis is conducted exclusively on the QM9 dataset due to its ample volume of test data, ensuring a statistically significant number of samples within each defined bin for reliable outcomes. Initially, the dataset is partitioned into training, validation, and test sets using scaffold splits. We then compute the average Tanimoto similarity between each test data point and the unique training scaffolds (1,096 in total). Test data points are sorted and grouped into 5 bins based on descending similarity scores, with the most similar points placed in the first bin. The average Tanimoto similarities for these 5 bins are 0.116, 0.103, 0.093, 0.084, and 0.066, respectively. During the experiments, each combination of backbone and UQ

[1] https://moleculenet.org/

[2] https://github.com/dptech-corp/Uni-Mol/tree/main/unimol

model is trained once on the training dataset and then tested separately across each of the bins. This process is repeated three times using random seeds 0, 1, and 2, with the results averaged across these runs. For the Deep Ensembles method, these three runs are integrated into a single ensemble, considered equivalent to a single run.

A.2 Backbone Models and Implementation Details

Our experimental code is designed on the PyTorch framework [207]. The experiments are conducted on a single NVIDIA A100 Tensor Core GPU with a memory capacity of 80GB. The backbone models are fine-tuned using the AdamW optimizer [208] with a weight decay rate of 0.01 using full-precision floating-point numbers (FP32) for maximum compatibility. We apply different learning rates, numbers of training epochs and batch sizes for the backbone models, as specified in the following paragraphs. We adopt early stopping to select the best-performed checkpoint on the validation set, and all models have achieved their peak validation performance before the training ends. ROC-AUC is selected to assess classification validation performance. For regression, we follow [67] and use RMSE for Physical Chemistry properties and MAE for Quantum Mechanics. Notice that these metrics only concern prediction, and do not take the uncertainty into account during validation steps. Table 2.1 presents the number of parameters and average time per training step for each backbone model. The training time is calculated for each model update step instead of each epoch based on our implementation, which might be slower than the models' original realizations. Below, we offer a detailed description of each backbone model's architecture and its implementation.

ChemBERTa ChemBERTa [26, 35] leverages the RoBERTa model architecture and pre-training strategy [209] but with fewer layers, attention heads, and smaller hidden dimensionalities. Unlike language models which process sentences in natural language, ChemBERTa uses Simplified SMILES [34] strings as input. This representation is a compact and linear textual depiction of a molecule's structure that's frequently employed in cheminformatics. ChemBERTa pre-training adopts a corpus of 77M SMILES strings from PubChem [210], along with the MLM objective [10].

ChemBERTa is built with the HuggingFace Transformers library [211], and its pre-trained parameters are shared through the Huggingface's model hub. We retain the default model architecture and use the `DeepChem/ChemBERTa-77M-MLM` checkpoint for ChemBERTa's weight initialization.[3] We employ the last-layer hidden state corresponding to the token `[CLS]` to represent the input SMILES sequence and attach the output heads specified by the tasks/UQ methods on top of it for property prediction. On all datasets, ChemBERTa is fine-tuned with a learning rate of 10^{-5}, a batch size of 128, and for 200 epochs. A tolerance of 40 epochs for early stopping is adopted.

GROVER GROVER [36] is pre-trained on 11 million unlabeled molecules represented as 2D molecular graphs, sourced from ZINC15 [212] and ChEMBL [213]. This model enhances pre-training by integrating MPNN [84], a GNN-based method, into Transformer encoder architectures. GROVER employs self-supervised learning objectives that reflect different levels of molecular structural complexity to capture detailed structural and semantic information. Specifically, it replaces the linear layers that map the input query, key, and value vectors in each multi-head attention module of the Transformer encoder with a dynamic MPNN, which aggregates latent features from k-hop neighboring nodes, where k is a random integer chosen from a predefined uniform or Gaussian distribution. In essence, GROVER integrates GNN layers before each head's self-attention process to more effectively encode the molecular graph structure. To aid learning, GROVER introduces training objectives such as contextual property prediction, which estimates the statistical properties of masked subgraphs of varying sizes, and graph-level motif prediction, which identifies the classes of masked functional groups. A readout function is used to aggregate node features, which are subsequently processed by linear layers for property prediction.

[3] https://huggingface.co/DeepChem/ChemBERTa-77M-MLM.

For implementation, we use the GROVER-base checkpoint for our model initialization.[4] We incorporate the "node-view" branch as discussed above and disregard the "edge-view" architecture detailed in the appendix of [36], in accordance with the default settings in their GitHub repository. Under this configuration, the model generates 2 sets of node embeddings (but no edge embeddings), one from the node hidden states and another from the edge hidden states [36]. Each set of embeddings is passed into 2 linear layers with GELU [214] activation and a dropout ratio of 0.1 post-readout layer, which simply averages the embeddings in the default configuration. Each set of embeddings corresponds to an individual output branch, predicting the properties independently. In line with the original implementation, we compute the loss for each branch individually during fine-tuning and apply a squared Euclidean distance regularization with a coefficient of 0.1 between the two. During inference, we average the logits from the 2 branches to generate the final output logits. Our implementation of GROVER does not incorporate external RDKit or Morgan fingerprints, diverging from the authors' original implementation.

In our experiments, we configure the fine-tuning batch size at 256, and the number of epochs at 100 with a tolerance of 40 epochs for early stopping. The learning rate is set at 10^{-4}, and the entire model has a dropout ratio of 0.1. We substitute the original Noam learning rate scheduler [13] with a linear learning rate scheduler with a 0.1 warm-up ratio for easier implementation. No substantial differences in model performance were observed between the two learning rate schedulers.

Uni-Mol Uni-Mol [29] is a universal molecular representation framework that enhances representational capacity and broadens applicability by incorporating 3D molecular structures as model input. For the property prediction task, Uni-Mol undergoes pre-training on 209M 3D conformations of organic molecules gathered from ZINC15 [212], ChEMBL [213], and a database comprising 12M purchasable molecules [29]. It portrays atoms as tokens and utilizes pair-type aware Gaussian kernels to encode the positional information

[4] https://github.com/tencent-ailab/grover

in the 3D space, thereby ensuring rotational and translational invariance. Furthermore, Uni-Mol introduces a pair-level representation by orchestrating an “atom-to-pair” communication, updating positional encodings with query-key products, and a “pair-to-atom” communication, adding pair representation as bias terms in the self-attention atom update procedure. For pre-training, Uni-Mol employs masked atom prediction, akin to BERT’s MLM, corrupting 3D positional encodings with random noise at a 15% ratio. Additionally, the model is tasked with restoring the corrupted Euclidean distances between atoms and the coordinates for atoms.

Our codebase is developed atop the publicly accessible Uni-Mol repository and their pre-trained checkpoint for molecular prediction serves as our model initialization. During fine-tuning, Uni-Mol generates 10 sets of 3D conformations for each molecule, supplemented with an additional 2D molecular graph. Thereafter Uni-Mol samples one from these 11 molecular representations for each molecule at the beginning of every training epoch as the input feature. For inference, we average the logits from all 11 representations to generate the final output. We utilize the conformations prepared by the Uni-Mol repository in our implementation.

We configure the fine-tuning batch size at 128, the number of epochs at 100, and employ early stopping with a tolerance of 40 epochs. We use a linear learning rate scheduler with a 0.1 warm-up ratio and a peak learning rate of 5×10^{-5}. The model is trained with a dropout ratio of 0.1. Although the Uni-Mol repository provides a set of recommended hyperparameters, we observe no discernible improvement in model performance with these settings.

DNN DNN serves as a simple, randomly initialized baseline model designed to explore how heuristic descriptors like Morgan fingerprints [215] or RDKit features perform for molecular property prediction. DNN enables us to compare the pre-trained models, which learn the molecular representation automatically through self-learning, with heuristic molecular features, which are constructed manually, and investigate whether or under what circumstances the heuristic features can achieve comparable results. For the descriptor, we adopt the approach of previous work [67, 216, 36] and extract 200-dimensional molecule-level features using RDKit for each molecule, which are then used as DNN input.[5] The DNN consists of 8 fully connected 128-dimensional hidden layers with GELU activation and an intervening dropout ratio of 0.1. We find no performance gain from deeper or wider DNNs and thus assume that our model is fully capable of harnessing the expressivity of RDKit features. The model is trained with a batch size of 256 and a constant learning rate of 2×10^{-4} for 400 epochs with an early stopping tolerance of 50 epochs.

TorchMD-NET The architectural design of TorchMD-NET is detailed in [92], while its pre-training methodology is discussed in a separate study [93]. TorchMD-NET is an equivariant Transformer tailored for the prediction of quantum mechanical properties of 3D conformers. Unique elements of its architecture include a specialized embedding layer—encoding not just atomic numbers but also the interatomic distances influenced by neighboring atoms, a modified multi-head attention mechanism that integrates edge data, and an equivariant update layer computing atomic interactions. The model undergoes pre-training on the 3.4M PCQM4Mv2 dataset [217, 218], leveraging a denoising auto-encoder objective. This entails predicting Gaussian noise disturbances on atomic positions, mirroring techniques seen in prevalent diffusion models in computer vision [219].

To implement TorchMD-NET, we sourced the code and model checkpoint from [93].[6] We made minor architectural adjustments, replacing their single-head output block with our adaptive multi-head output layers. Consequently, we omitted the denoising objective during the fine-tuning process due to compatibility concerns. Our fine-tuning regimen for TorchMD-NET entails a batch size of 128 over 100 epochs, adopting an early stopping mechanism with the patience of 40 epochs. The learning rate peaks at 2×10^{-4}, coupled

[5] We use the `rdkit2dnormalized` descriptor in `DescriptaStorus`, available at https://github.com/bp-kelley/descriptastorus.

[6] https://github.com/shehzaidi/pre-training-via-denoising

Table A.1: Impact of the number and dimensionality of hidden layers across various datasets. The selected hyper-parameters generally achieve reasonable performance, although they may not be optimal for all datasets.

	Number of Layers (Dimension: 128)				Layer Dimensionality (Number of Layers: 5)			
Dataset: Metric	3 Layers	4 Layers	5 Layers	6 Layers	64 Dim	128 Dim	256 Dim	512 Dim
BBBP: ROC-AUC ↑	0.5736	0.6233	0.6284	0.6223	0.5876	0.6284	0.6255	0.5977
Tox21: ROC-AUC ↑	0.6644	0.6774	0.6832	0.6901	0.6584	0.6832	0.6924	0.6962
ESOL: RMSE ↓	1.6741	1.664	1.6017	1.6383	1.9951	1.6017	1.6167	1.5868
FreeSolv: RMSE ↓	2.3296	2.2348	2.1362	2.5788	4.0666	2.1362	2.8263	3.1005

with a linear scheduler with a 0.1 warm-up ratio, and the model trains with a dropout ratio of 0.1.

GIN Graph Isomorphism Network GIN [94] is a randomly initialized model with 2D graph structures as input. Compared with Graph Convolutional Networks (GCNs) [220], GIN mainly differs in that within the neighboring nodes message aggregation process, GIN adds a weight to each node's self-looping, which is either trainable or pre-defined. In addition, GIN substitutes the one-layer feed-forward network within each GCN layer with a MLP. It has been proved in theory that these minor changes make GIN among the most powerful graph neural networks [94].

We use the Pytorch Geometric [221] to realize GIN. Our implementation contains 5 GIN layers with 128 hidden units and 0.1 dropout ratio. A study of the hyperparameters of GIN is presented in Table A.1. The model is trained with a batch size of 128 for 200 epochs with an early stopping tolerance of 50 epochs, at a constant learning rate of 10^{-4}.

A.3 Uncertainty Quantification

A.3.1 Method and Implementation Details

Focal Loss First proposed by [95], Focal Loss is designed to address the class imbalance issue for dense object detection in computer vision, where the number of negative samples (background) far exceeds the number of positive ones (objects). It is adopted for uncertainty estimation and model calibration later by [96]. The idea is to add a modulating factor to the standard cross-entropy loss to down-weight the contribution from easy examples and thus focus more on hard examples. In the binary classification case, it adds a modulating factor $|y_n - \hat{p}_n|^\gamma$ to the standard cross-entropy loss, where $y_n \in \{0, 1\}$ is the ground truth label, $\hat{p}_n \in (0, 1)$ is the predicted Sigmoid probability for the n-th example, and $\gamma \geq 0$ is a focusing parameter:

$$\mathcal{L}_{\text{focal}} = -\frac{1}{N} \sum_{n=1}^{N} \left[y_n (1 - \hat{p}_n)^\gamma \log \hat{p}_n + (1 - y_n) \hat{p}_n^\gamma \log(1 - \hat{p}_n) \right] . \tag{A.1}$$

The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, Focal Loss is equivalent to the cross-entropy loss. As γ increases, the effect of the modulating factor increases likewise. In our implementation, we take advantage of the realization in the `torchvision` library and use their default hyperparameters for all experiments.[7]

Bayes by Backprop BBP [97, 222] is an algorithm for training BNNs, where weights are not point estimates but distributions. The idea is to replace the deterministic network weights with Gaussian a porteriori learned from the data, which allows quantifying the uncertainty in the predictions by assembling the predictions from random networks sampled from the posterior distribution of the weights:

$$\boldsymbol{W}^{\text{MAP}} = \arg\max_{\boldsymbol{W}} \log p(\boldsymbol{W}|\mathbb{D}) = \arg\max_{\boldsymbol{W}} \left(\log p(\mathbb{D}|\boldsymbol{W}) + \log p(\boldsymbol{W}) \right) , \tag{A.2}$$

where $p(\boldsymbol{W})$ is the prior distribution of the weights, which are also Gaussian in our realization.

However, the true posterior is generally intractable for neural networks and can only be approximated with variational inference $q(\boldsymbol{W}|\boldsymbol{\theta})$ [98], where $\boldsymbol{\theta}$ are variational parameters,

[7] https://pytorch.org/vision/main/generated/torchvision.ops.sigmoid_focal_loss.html.

which consist of the multivariate Gaussian mean and variance in our case. The learning then involves finding the $\boldsymbol{\theta}$ that minimizes the KLD between the true posterior and the variational distribution. The loss can be written as

$$\mathcal{L} = D_{\mathrm{KL}}\left(q(\boldsymbol{W}|\boldsymbol{\theta})||p(\boldsymbol{W})\right) - \mathbb{E}_{q(\boldsymbol{W}|\boldsymbol{\theta})}[\log p(\mathbb{D}|\boldsymbol{W})]. \tag{A.3}$$

For each training step i, we first draw a sample from a porteriori $\boldsymbol{W}_i \sim q(\boldsymbol{W}|\boldsymbol{\theta})$ and then compute the Monte Carlo estimation of the loss:

$$\mathcal{L}_i \approx \log q(\boldsymbol{W}_i|\boldsymbol{\theta}) - \log p(\boldsymbol{W}_i) - \log p(\mathbb{D}|\boldsymbol{W}_i). \tag{A.4}$$

During backpropagation, the gradient can be pushed through the sampling process with the reparameterization trick [98]. Specifically, we adopt the local reparameterization trick [222], which samples the pre-activation $\boldsymbol{a}_i$ directly from the distribution $q(\boldsymbol{a}_i|\boldsymbol{x}_i)$ parameterized by the input feature $\boldsymbol{x}_i$, instead of computing the is as $\boldsymbol{a}_i = \boldsymbol{W}_i\boldsymbol{x}_i$ using the sampled network weights. This has parameters that are deterministic functions of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}$, which reduces the variance of the gradient estimates and can improve the efficiency of the learning process. Following the common practice [223], we only apply BBP to the models' output layer to reduce computational costs and optimization difficulty for large pre-trained backbone models. In addition, The last layer often directly relates to the task's uncertainty. Modeling uncertainty in this layer can provide practical benefits in decision-making processes, allowing for an estimation of confidence in the predictions. During inference, we sample 30 networks from the posterior distribution, generating 30 sets of prediction results, and computing the mean of the predictions as the final output.

SGLD SGLD [99] combines the efficiency of SGD with Langevin diffusion which introduces the ability to estimate parameter a posteriori. The update rule for SGLD is given by:

$$\Delta\boldsymbol{\theta} = -\frac{\eta_t}{2}\nabla\mathcal{L}(\boldsymbol{\theta}) + \sqrt{\eta_t}\boldsymbol{\epsilon}, \tag{A.5}$$

where $\boldsymbol{\theta}$ is the network parameters, η_t is the learning rate at time t, and $\boldsymbol{\epsilon}_t$ is the standard Gaussian noise. With learning rate $\boldsymbol{\theta}$ or weight gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$ decreasing to small values, the update rule can transit from network optimization to posterior estimation. After sufficient optimization, subsequent samples of parameters can be seen as drawing from the posterior distribution of the model parameters given the data.

In our implementation, we follow the previous implementation and use a constant learning rate.[8] Similar to BBP, we only apply SGLD to the last layer of the model. We first train the model until its performance has stopped improving on the validation set, and then continue training it for another 30 epochs, resulting in 30 networks sampled from the Langevin dynamics. This generates 30 sets of prediction results during the test, and we compute the mean of the predictions as the final output.

MC Dropout Compared to other Bayesian networks, MC Dropout [79] is a simple and efficient for modeling the network uncertainty. Dropout is proposed to prevent overfitting by randomly setting some neurons' outputs to zero during training [224]. At test time, dropout is deactivated and the weights are scaled down by the dropout rate to simulate the presence of all neurons. In contrast, MC Dropout proposes to keep dropout active during testing and make predictions with dropout turned on. By running several (*e.g.*, 30 in our experiments) forward passes with random dropout masks, we effectively obtain a Monte Carlo estimation of the predictive distribution.

SWAG SWAG [101] is an extension of the Stochastic Weight Averaging [102] method, a technique used for finding wider optima in the loss landscape and leads to improved generalization. SWAG fits a Gaussian distribution with a low rank plus diagonal covariance

[8] https://github.com/JavierAntoran/Bayesian-Neural-Networks/.

derived from the SGD iterates to approximate the posterior distribution over neural network weights. In SWAG, the model keeps tracking the weights encountered during the last T steps of the stochastic gradient descent updates and computes the Gaussian mean and covariance:

$$\begin{aligned}\boldsymbol{\mu}_{\boldsymbol{\theta}} &= \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\theta}_t;\\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}} &= \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{\theta}_t - \boldsymbol{\mu}_{\boldsymbol{\theta}})(\boldsymbol{\theta}_t - \boldsymbol{\mu}_{\boldsymbol{\theta}})^{\mathsf{T}}.\end{aligned} \tag{A.6}$$

However, such computation requires storing all the model weights in the last T steps, which is expensive for large models. To address this issue, [101] propose to approximate the covariance matrix with a low-rank plus diagonal matrix, and compute the mean and covariance iteratively. Specifically, at update step $t \in \{1, \ldots, T\}$,

$$\bar{\boldsymbol{\theta}}_t = \frac{t\bar{\boldsymbol{\theta}}_{t-1} + \boldsymbol{\theta}_t}{t+1}; \quad \overline{\boldsymbol{\theta}^2}_t = \frac{t\overline{\boldsymbol{\theta}^2}_{t-1} + \boldsymbol{\theta}_t^2}{t+1}; \quad \widehat{D}_{:,t} = \boldsymbol{\theta}_t - \bar{\boldsymbol{\theta}}_t, \tag{A.7}$$

where $\bar{\boldsymbol{\theta}}_0$ is the best parameter weights found during training prior to the SWA session, $\overline{\boldsymbol{\theta}^2}_T - \bar{\boldsymbol{\theta}}_T^2$ is the covariance diagonal, and $\frac{1}{T-1}\widehat{D}\widehat{D}^{\mathsf{T}} \in \mathbb{R}^{d,d}$ is the low-rank approximation of the covariance matrix with d being the parameter dimensionality. We can write the Gaussian weight posterior as

$$\boldsymbol{\theta}_{\text{SWAG}} \sim \mathcal{N}_{\text{SWAG}}\left(\bar{\boldsymbol{\theta}}_T, \frac{1}{2}\left(\overline{\boldsymbol{\theta}^2}_T - \bar{\boldsymbol{\theta}}_T^2 + \frac{1}{T-1}\widehat{D}\widehat{D}^{\mathsf{T}}\right)\right). \tag{A.8}$$

Notice that $\hat{D}$ has a different rank $K <= T$ in [101], but we set $K = T < d$ for simplicity. Uncertainty in SWAG is estimated by drawing weight samples from $\mathcal{N}_{\text{SWAG}}$ and running these through the network. We set $T = 20$, and draw 30 samples during the test in our experiments.

Temperature Scaling TS [82, 39] is a simple and effective post-hoc method for calibrating the confidence of a neural network. Post-hoc methods calibrate the output probabilities of a pre-trained model without updating the fine-tuned network parameters. The core idea behind TS is to add a learnable parameter h (the temperature) to adjust the output probability of the model. For a trained binary classification model, TS scales the logits z with

$$z' = \frac{z}{h} \tag{A.9}$$

before feeding z' into the Sigmoid output activation function. The temperature h is learned by minimizing the negative log-likelihood of the training data with other network parameters frozen. For multi-task classification such as Tox21, we assign an individual temperature to each task.

In precise terms, "Temperature Scaling" is introduced for multi-class classification utilizing SoftMax output activation [39]. For binary classification in our study, we implement Platt Scaling, excluding the bias term [82]. Nonetheless, we continue using "Temperature Scaling" (TS) for its widespread recognition.

Deep Ensembles Deep Ensembles [80] is a technique where multiple deep learning models are independently trained from different initializations, and their predictions are combined to make a final prediction. This approach exploits the idea that different models will make different types of errors, which can be reduced by averaging model predictions, leading to better overall performance and more robust uncertainty estimates [104]. Formally, given M models in the ensemble, each with parameters $\boldsymbol{\theta}_m, m \in \{1, \ldots, M\}$, the ensemble prediction for an input data point $\boldsymbol{x}$ is given by:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m = \frac{1}{M} \sum_{m=1}^{M} f(\boldsymbol{x}; \boldsymbol{\theta}_m) \tag{A.10}$$

where f represents the model architecture, and $\hat{y}_m$ is the post-activation result of the m-th model. We set $M = 3$ for QM8, QM9 and MUV to reduce computational consumption, and $M = 10$ for other scaffold split datasets. For random split, we uniformly use $M = 3$.

For regression tasks, [80] aggregate the variances of different network predictions through parameterizing a Gaussian mixture model. In contrast, we take a simpler approach by computing the mean of the variances as the final output variance.

A.3.2 Resource Analysis

Table 2.2 summarizes the additional training cost to apply each UQ method to a backbone model already fine-tuned for property prediction. From the table, we can see that post-hoc calibration and MC Dropout are the most efficient methods, while Deep Ensembles is undoubtedly the most expensive one, even though it performs the best most of the time. Several works aim to reduce the computational cost [225, 226], but we do not consider them in MUBen and leave them to future works.

Appendix B

Language Model Uncertainty Quantification

B.1 Notations

Table B.1 and Table B.2 summarizes the notations and definitions used in this chapter.

B.2 UQAC Details

B.2.1 Attention Re-Weighting

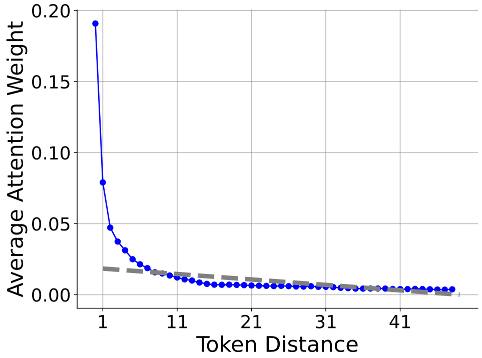


Figure B.1: Averaged attention weights.

As discussed in subsection 3.3.1, to mitigate the inherent bias of LLMs toward emphasizing more recent context, we apply a sequence of decreasing re-weighting factors $\boldsymbol{\gamma} \in [0,1]^C$ to adjust the attention weights corresponding to the most recent C tokens. To obtain these re-weighting factors, we first compute an average attention vector $\overline{\boldsymbol{\alpha}}$. Specifically, we randomly select 1,000 samples

Table B.1: Summary of Notations and Definitions (Continued in Table B.2)

Category	Notation	Definition
Sets & Spaces	$\mathbb{V}$	Vocabulary set.
	$\mathbb{V}_{\text{stop}}$	Set of stop words.
	$\mathbb{N}_{[1,N]}$	Set of natural numbers from 1 to N.
	$\mathbb{S}$	Reduced reasoning space.
	$\emptyset$	Empty set.
Tokens & Sequences	x	A sampled token.
	x	A random token variable.
	$\boldsymbol{x}$	A sampled token sequence/vector.
	$\mathbf{x}$	A random sequence variable.
	$\boldsymbol{x}_{\text{instr}}$	Instruction sequence.
	$\boldsymbol{x}_{\text{resp}}$	Response sequence.
	$\boldsymbol{x}_{\text{cot}}$	Reasoning (CoT) sequence.
	$\boldsymbol{x}_{\text{ans}}$	Answer token sequence.
	$\boldsymbol{x}_{\text{src}}^{(z)}$	Source sequence/set at backtracking step z.
	$\boldsymbol{x}_{\text{tgt}}^{(z)}$	Target sequence/set at backtracking step z.
	$\boldsymbol{x}_{\text{attn}}$	Attention chain token sequence.
	$\boldsymbol{x}'_{\text{attn}}$	Similarity-filtered attention chain sequence.
Attention Bachtracking	$\boldsymbol{\alpha}_T^{(l,h)}$	Attention weights at step T in layer l, head h.
	$\boldsymbol{\alpha}'$	Re-weighted attention weights.
	$\boldsymbol{\alpha}_T^*$	Aggregated attention weights at step T.
	$\boldsymbol{\gamma}$	Re-weighting factors.
	$\boldsymbol{\varphi}$	Cumulative attention weights from all source tokens.
	$\boldsymbol{w}$	Similarity vector.
Probabilistic Metrics	P	Exact probability.
	$\widetilde{P}$	Approximated probability.
	$\overline{P}$	Length-normalized approximated probability.
	$\mathcal{H}$	Entropy measure.

Table B.2: Summary of Notations and Definitions (Continued)

Category	**Notation**	**Definition**
Numbers & Indices	L_{instr}	Length of the instruction sequence.
	L_{resp}	Length of the response sequence.
	L_{cot}	Length of the reasoning sequence.
	L_{ans}	Length of the answer sequence.
	L_{attn}	Length of the attention chain.
	L'_{attn}	Length of the similarity-filtered attention chain.
	L_{tgt}	Length of the target sequence/set.
	t	Token index in a sequence.
	T	Index of the latest predicted token during inference.
	i, j, k	Generic indices (context-dependent).
	l, h	Layer and head indices in model $\mathcal{M}$.
	L, H	Total number of layers and heads in model $\mathcal{M}$.
	z	Attention backtracking step index.
	s, e	Position indices in source/target sequences or sets.
	t_s, t_e	Specific token positions in source and target sequences.
	C	Number of tokens selected for re-weighting.
	θ	Attention weight threshold.
Functions	$f(\cdot)$	Step-wise attention backtracking function.
	$\mathbb{I}(\cdot)$	Indicator function.
	$\lvert \cdot \rvert$	Cardinality of a set/sequence/vector.
	$\lVert \cdot \rVert$	2-norm of a vector.
	$\text{sim}(\cdot, \cdot)$	Cosine similarity.
Operators	$\oplus$	Concatenation of sequences.
	$\sqsubset$	Proper subsequence relation (non-consecutive allowed).
	$\sqsubseteq$	Subsequence relation.
	$\cdot^{\mathsf{T}}$	Transpose of a vector or matrix.
	$\cdot^{T}$	Exponentiation by T.
	$\cdot^{(T)}$	Superscript T.
	$\arg\min_k \boldsymbol{a}$	Index of the k-th smallest element in vector $\boldsymbol{a}$.
	$\arg\max_k \boldsymbol{a}$	Index of the k-th largest element in vector $\boldsymbol{a}$.
	$\boldsymbol{a}_i$	A vector from $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \dots\}$ indexed by subscript i.
	$a_{i,[j]}$	j-th element in the vector $\boldsymbol{a}_i$.
	$\boldsymbol{a}_{i,[\backslash j]}$	Subvector/subset of $\boldsymbol{a}_i$ excluding the j-th element.
	$\boldsymbol{a}_{i,[j,k]}$	Subvector of $\boldsymbol{a}_i$ from the j-th to k-th elements (inclusive).

from the GSM8k training partition and use Llama-3.2-1B to generate answers for each sample. We then average the attention weights from all heads and layers for the last 50 tokens in each generated response:

$$\overline{\boldsymbol{\alpha}} \in [0,1]^{50} = \frac{1}{HLL_{\text{resp}}} \sum_{h=1}^{H} \sum_{l=1}^{L} \sum_{t=L_{\text{instr}}+1}^{L_{\text{instr}}+L_{\text{resp}}} \boldsymbol{\alpha}_{t,[t-50:t]}^{(h,l)}. \tag{B.1}$$

To simplify subsequent analysis, we reverse the order of the computed average attention weights, placing the most recent token at the first position, *i.e.*, $\overline{\alpha}_{[i]} \leftarrow \overline{\alpha}_{[50-i]}$. Next, we project $\overline{\alpha}$ into a two-dimensional space, with the y-axis representing attention weights and the x-axis representing token indices. We then fit a linear regression line that passes through the points $(C+1, \overline{\alpha}_{[C+1]})$ and $(\lfloor (50-C)/2 \rfloor, \overline{\alpha}_{[\lfloor (50-C)/2 \rfloor]})$. This regression line is defined as:

$$g(i) = \overline{\alpha}_{[C+1]} + \frac{\overline{\alpha}_{[\lfloor \frac{50-C}{2} \rfloor]} - \overline{\alpha}_{[C+1]}}{\lfloor \frac{50-C}{2} \rfloor - (C+1)} (i - (C+1)). \tag{B.2}$$

Subsequently, each re-weighting factor $\gamma_{[i]}$ is calculated as:

$$\gamma_{[C-i+1]} = \frac{g(i)}{\overline{\alpha}_{[i]}} \quad \forall i \in \mathbb{N}_{[1,C]}. \tag{B.3}$$

We emphasize that the resulting $\boldsymbol{\gamma}$ values are approximations, as high precision is not critical for this adjustment. Consequently, we adopt these GSM8k and Llama-3.2-1B-derived factors universally across all models and datasets. In our experiments, we consistently set

$C = 10$, yielding the following re-weighting factors:

$$\gamma = \begin{bmatrix} 0.93925344 \\ 0.87378443 \\ 0.81274293 \\ 0.73914525 \\ 0.67549127 \\ 0.59304059 \\ 0.46061748 \\ 0.32959151 \\ 0.20938152 \\ 0.16644488 \end{bmatrix}, \tag{B.4}$$

which is hard-coded in our implementation. The averaged attention weights and the fitted line are visualized in Figure B.1.

B.3 Datasets and Processing

Our implementation separates answer generation from the uncertainty quantification step. Once an answer is generated, we first extract and evaluate it to verify its correctness. If no answer is extracted, the instance is excluded from the UQ evaluation. Next, we subsample the instances to balance the number of correct and incorrect predictions, ensuring a reliable estimation of the calibration metrics. Specifically, if the number of correct predictions is lower than that of incorrect ones, we randomly sample a matching number of incorrect predictions. If both groups exceed 500 instances, we randomly select 500 instances from each group. Although this step does not affect the UQ scores, it does influence both the AUROC and ECE metrics. Therefore, we repeat the subsampling process five times and report the mean and standard deviation of the results.

Appendix C

Weakly Supervised Named Entity Recognition

C.1 Training Objective

In this section, we focus on the computation of the expected complete data log likelihood Q defined in (Equation 4.14) as well as the training objective. We skip some trivial steps and explanations. (Equation 4.14) can be write as:

$$
\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{i=1}^{L} p(z^{(0)} = i | \boldsymbol{x}^{(1:T)}, \boldsymbol{e}^{(0:T)}) \log p(z^{(0)} = i) + \\
\sum_{t=1}^{T} \sum_{i=1}^{L} \sum_{j=1}^{L} p(z^{(t-1)} = i, z^{(t)} = j | \boldsymbol{x}^{(1:T)}, \boldsymbol{e}^{(0:T)}) \log \Psi_{i,j}^{(t)} + \\
\sum_{t=1}^{T} \sum_{i=1}^{L} p(z^{(t)} = i | \boldsymbol{x}^{(1:T)}, \boldsymbol{e}^{(0:T)}) \log \varphi_{i}^{(t)},
\end{aligned}
\tag{C.1}
$$

where $\boldsymbol{\Psi}$ is the transition matrix. $p(z^{(0)})$ is the probability of the initial hidden state without any corresponding observations. As we can predict the token-wise transition matrix from the embeddings, we can simply set it to Uniform or, as [59, 60] proposed, set $p(z^{(0)} = 1)$ to 1 and $p(z^{(0)} = i), \forall i \in 2 : L$ to 0.

To calculate (Equation C.1), we define the smoothed marginal $\boldsymbol{\gamma}^{(t)} \in [0,1]^L$ as:

$$\gamma_i^{(t)} \triangleq p(z^{(t)} = i | \boldsymbol{x}^{(1:T)}, \boldsymbol{e}^{(0:T)}),$$

and the expected number of transitions $\boldsymbol{\xi}^{(t)} \in [0,1]^{L \times L}$ as:

$$\xi_{i,j}^{(t)} \triangleq p(z^{(t-1)} = i, z^{(t)} = j | \boldsymbol{x}^{(1:T)}, \boldsymbol{e}^{(0:T)}).$$

These two variables are acquired using the *forward-backward* algorithm.

First, we define the filtered marginal $\boldsymbol{\alpha} \in [0,1]^L$ as:

$$\alpha_i^{(t)} \triangleq p(z^{(t)} = i | \boldsymbol{x}^{(1:t)}, \boldsymbol{e}^{(0:T)}),$$

and the conditional future evidence $\boldsymbol{\beta} \in [0,1]^L$ as:

$$\beta_i^{(t)} \triangleq p(\boldsymbol{x}^{(t+1:T)} | z^{(t)} = i, \boldsymbol{e}^{(0:T)}).$$

In the forward pass, $\alpha_i^{(t)}$ is computed iteratively:

$$\begin{aligned} \alpha_i^{(t)} &\propto p(\boldsymbol{x}^{(t)} | z^{(t)} = i, \boldsymbol{e}^{(0)}) p(z^{(t)} = i | \boldsymbol{x}^{(1:t-1)}, \boldsymbol{e}^{(0:t)}) \\ &= \sum_{j=1}^{L} \varphi_i^{(t)} \Psi_{j,i}^{(t)} \alpha_j^{(t-1)}, \end{aligned}$$

which can be written in the matrix form:

$$\boldsymbol{\alpha}^{(t)} \propto \boldsymbol{\varphi}^{(t)} \odot ({\boldsymbol{\Psi}^{(t)}}^{\mathsf{T}} \boldsymbol{\alpha}^{(t-1)}),$$

where $\odot$ is the element-wise product. We initialize $\boldsymbol{\alpha}$ with $\alpha_l^{(0)} = p(z^{(0)} = l), \forall l \in 1 : L$ since we have no observation at time step 0.

Similarly, we do the backward pass and compute $\boldsymbol{\beta}$:

$$\begin{aligned}\beta_i^{(t-1)} &= \sum_{j=1}^{L} p(z^{(t)} = j, \boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1:T)} | z^{(t-1)} = i, \boldsymbol{e}^{(0,t:T)}) \\ &= \sum_{j=1}^{L} \beta_j^{(t)} \varphi_j^{(t)} \Psi_{i,j}^{(t)}.\end{aligned}$$

In the matrix form, it becomes:

$$\boldsymbol{\beta}^{(t-1)} = \boldsymbol{\Psi}^{(t)}(\boldsymbol{\varphi}^{(t)} \odot \boldsymbol{\beta}^{(t)}),$$

with base case:

$$\beta_i^{(T)} = p(\boldsymbol{x}^{(T+1:T)} | z^{(T)} = i) = 1, \forall i \in 1 : L.$$

With $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ calculated, $\gamma_i^{(t)}$ and $\xi_{i,j}^{(t)}$ can be written as:

$$\begin{aligned}\gamma_i^{(t)} &\propto p(z^{(t)} = i | \boldsymbol{x}^{(1:t)}, \boldsymbol{e}^{0:t}) p(\boldsymbol{x}^{(t+1:T)} | z^{(t)} = i, \boldsymbol{e}^{(0,t+1:T)}) \\ &= \alpha_i^{(t)} \beta_i^{(t)}, \\ \xi_{i,j}^{(t)} &\propto p(z^{(t-1)} = i | \boldsymbol{x}^{(1:t-1)} \boldsymbol{e}^{(0:t-1)}) p(\boldsymbol{x}^{(t)} | z^{(t)} = j, \boldsymbol{e}^{(0)}) \\ &\quad p(\boldsymbol{x}^{(t+1:T)} | z^{(t)} = j, \boldsymbol{e}^{(0,t+1:T)}) p(z^{(t)} = j | z^{(t-1)} = i, \boldsymbol{e}^{(t)}) \\ &= \alpha_i^{(t-1)} \varphi_j^{(t)} \beta_j^{(t)} \Psi_{i,j}^{(t)}.\end{aligned}$$

Written in the matrix form, they become:

$$\begin{aligned}\boldsymbol{\gamma}^{(t)} &\propto \boldsymbol{\alpha}^{(t)} \odot \boldsymbol{\beta}^{(t)}, \\ \boldsymbol{\xi}^{(t)} &\propto \boldsymbol{\Psi}^{(t)} \odot (\boldsymbol{\alpha}^{(t-1)} (\boldsymbol{\varphi}^{(t)} \odot \boldsymbol{\beta}^{(t)})^{\mathsf{T}}).\end{aligned}$$

Eventually, we insert $\boldsymbol{\gamma}$ and $\boldsymbol{\xi}$ into (Equation C.1) to compute the value of Q. The training objective is to maximize Q, which can be readily done using the gradient ascend. Please refer to [59] for more details.

Appendix D

Ensembles of Low-Rank Expert Adapters

D.1 Datasets

To evaluate the effectiveness of ELREA, we conducted experiments across two distinct categories: 1) general language understanding and reasoning, and 2) mathematical reasoning. Each category utilizes its own dedicated training and evaluation datasets, as detailed in Table 5.1.

General Language Understanding and Reasoning For the first category, we followed the methodologies outlined in [62] and [227]. We employed a diverse combination of datasets for fine-tuning our model:

- **Flan V2** [163]: This comprehensive collection encompasses over 1,800 NLP tasks, combining numerous existing datasets with various data augmentations. The tasks cover a wide range of NLP problems, including question answering, summarization, translation, and sentiment analysis.
- **Chain-of-Thought (CoT)** [20, 163]: A subset of the Flan V2 collection, the CoT dataset includes tasks annotated with chain-of-thought reasoning steps. It emphasizes

the model's ability to generate intermediate reasoning processes, enhancing performance on complex tasks that require multi-step reasoning.

- **Dolly-15k** [201]: This curated dataset contains approximately 15,000 high-quality, human-generated prompt-response pairs designed specifically for instruction tuning of LLMs. Created by Databricks employees, it focuses on instruction-following capabilities across a variety of domains and task types.
- **OpenAssistant Conversations** [202]: A multilingual, human-generated, and human-annotated assistant-style conversation corpus featuring fully annotated conversation trees in different languages. For our experiments, we utilize only the supervised fine-tuning portion of this dataset, excluding any content related to reward modeling or reinforcement learning.

These datasets vary significantly in size, format, tasks, and domains, providing a comprehensive training ground for general language understanding and reasoning. Specifically, Flan V2 and CoT datasets contribute to the model's ability to handle a wide range of NLP tasks with enhanced reasoning capabilities, while Dolly-15k and OpenAssistant Conversations improve the model's instruction-following and conversational skills. In practice, we directly use the pre-processed dataset provided by [62], which consolidates these datasets into a unified format suitable for fine-tuning.[1]

For testing, we utilize two challenging benchmark datasets to evaluate the general reasoning and problem-solving abilities of our model:

- **Massive Multitask Language Understanding** (MMLU; [203]): MMLU is a comprehensive evaluation benchmark that assesses a model's knowledge and reasoning across 57 subjects, including humanities, sciences, social sciences, and more. The dataset consists of over 19,000 multiple-choice questions designed to mimic the difficulty of an average professional or college-level exam. Each question has four answer options, and the dataset provides only the correct answer without any accompanying

[1] Available at https://huggingface.co/datasets/princeton-nlp/less_data.

reasoning or explanation.

- **BIG-Bench Hard** (BBH; [127, 128]): BBH is a subset of the BIG-Bench, consisting of 23 tasks identified as particularly challenging for LLMs. The tasks cover a diverse range of domains such as logical reasoning, mathematics, commonsense reasoning, *etc.*. Unlike MMLU, BBH includes not only the correct answers but also detailed CoT reasoning annotations for each question. This allows for the assessment of a model's ability to perform complex reasoning and generate intermediate reasoning steps.

Both datasets predominantly feature difficult multiple-choice question-answering formats with diverse question types, and only a few require numerical responses. The inclusion of reasoning chains in BBH enables a more in-depth evaluation of the model's reasoning capabilities compared to MMLU, which focuses solely on the final answers. Importantly, there is no significant overlap between the training datasets and these test datasets, ensuring that the evaluation measures the model's ability to generalize to unseen tasks and domains. To facilitate the desired output formatting and to guide the model during inference, we provide up to three in-context examples from the validation subset of the BBH dataset and five examples from MMLU dataset. These examples serve as prompts to help the model understand the expected answer format and improve its performance on the evaluation tasks.

Mathematical Reasoning For the mathematical reasoning category, we developed the MATH-Combined dataset by integrating several existing mathematical problem-solving resources into a unified format analogous to the MATH dataset [51], including

- **GSM8K** [50]: A dataset containing 8,000 high-quality grade school math word problems that require multi-step reasoning to solve. Each problem includes a question and a detailed step-by-step solution.
- **MathQA** [205]: Originally a multiple-choice dataset derived from the AI2 Arithmetic and the DeepMind Mathematics datasets, MathQA consists of over 37,000 math word

problems across various topics. Each problem comes with a question, multiple-choice answers, and annotated solution programs.

- **SVAMP** [204]: A dataset designed to test the robustness of math word problem solvers by introducing subtle variations to existing problems. It contains 1,000 problems that require careful reasoning to avoid common pitfalls.
- **MATH** [51]: A collection of 12,500 challenging competition-level math problems covering subjects like algebra, geometry, calculus, and more. Each problem includes a question and a detailed solution formatted in LaTeX.

To create a consistent and unified dataset, we process the inputs from GSM8K, MathQA, and SVAMP to match the format of the MATH dataset. We utilize Claude 3 Sonnet [228] to reformulate the final answers into a specified format, specifically using the "`\boxed{}`" command to enclose final answers. For MathQA, which is originally in a multiple-choice format, we retain only the correct answers and reformat them into value prediction tasks. This standardization ensures that all problems across the datasets have a uniform presentation, facilitating knowledge transfer and model training. During the processing, the reformatted outputs generated are compared to the original answers to ensure accuracy. If the model fail to produce the correct answer after five attempts, those instances are discarded to maintain the dataset quality.

Unlike the first category of general language understanding and reasoning, the fine-tuning and test datasets in MATH-Combined are similarly distributed. This alignment allows us to gain insights into the effectiveness of selecting task-specific data for fine-tuning, as it enables us to assess how well the model performs on tasks that closely resemble its training data. To manage computational resources efficiently, we sub-sample the test instances to approximately 1,000 problems per dataset. Preliminary experiments show that it provides a representative enough evaluation of the model's performance while reducing the computational burden.

D.2 Model Configurations

Our primary experiments utilize Gemma-2b [206], which contains 2.5 billion network parameters, as the core framework for their relative efficiency in training and inference. Specifically, we employ the instruction-tuned variant `gemma-1.1-2b-it`, known for its efficiency in smaller-scale settings. We also conduct experiments with the larger and more advanced Gemma2 model `gemma-2-9b-it` [130] to investigate the impact of backbone model representativeness on the relative performance.[2] For the LoRA modifications, we default to a rank $r = 8$ across all linear layers in the model (*i.e.*, {`q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `down_proj`, `gate_proj`}), which count as about 0.39% of the total network parameters. In a separate experiment targeting the MATH-Combined dataset, we also explore the impact of increasing the rank to $r = 64$. The adapter's scaling factor α and dropout rate are consistently set to $\alpha = 4r$ and $p_{\text{dropout}} = 0.1$, respectively. The architecture for cluster-wise adapters $\mathcal{Q}_c$ mirrors that of the base adapter $\mathcal{Q}_{\text{base}}$ to streamline implementation. We typically set the gradient projection dimensionality to $d_{\text{proj}} = 8192$, but also include experiments with $d_{\text{proj}} = 512$ to investigate the impact of dimensionality reduction on model performance.

Due to license restrictions, we are unable to use LLaMA-series models [18] for our experiments.

D.3 Fine-Tuning

For both dataset categories, we fine-tune the base adapter $\mathcal{Q}_{\text{base}}$ for 2 epochs using the Adam optimizer, with an initial learning rate of 5×10^{-5} that linearly decays to zero. Preliminary testing indicates that 2 epochs optimize performance for $\mathcal{Q}_{\text{base}}$, ensuring a fair comparison with our method. We also observe a strong tendency toward overfitting beyond

[2] Available at https://huggingface.co/google/gemma-2-9b-it.

this point, as indicated by the loss value and gradient norm curve. Cluster-wise adapters $\mathcal{Q}_c$ undergo an identical duration of fine-tuning at a slightly reduced learning rate of 2×10^{-5}. These hyperparameters, derived from prior experience, are fixed without adjustments to preemptively accommodate unseen test data, diverging from the methods of [62]. Most fine-tuning sessions are conducted on an computing instance equipped with 8 NVIDIA A100 40GB GPUs, employing 4-bit quantization for the backbone model $\mathcal{M}$ and bf16 precision for adapters $\mathcal{Q}$. This setup essentially uses QLoRA [168] rather than LoRA, but we do not specifically distinguish them as they both belong to the LoRA family and do not impact our conclusions. Additional training sessions utilize instances with 8 NVIDIA V100 32GB GPUs, using fp16 precision. We observe no difference in performance between these configurations apart from training speed. The maximum token sequence length for training is 2,048, with a batch size of 16 sequences distributed across the GPU instances. Only a few (< 100 for each dataset category using the Gemma-2b tokenizer) of training sequences are longer than this threshold, and we simply discard these instances.

D.4 Baselines

Our primary baseline is the **base LoRA adapter** $\mathcal{M} + \mathcal{Q}_{\textbf{base}}$, which is fine-tuned on the complete dataset for 2 epochs to achieve optimal performance, as detailed in Section section D.3. Additionally, we consider a **dataset-wise adapter** $\mathcal{M} + \mathcal{Q}_{\textbf{dataset}}$ for MATH-Combined, where the adapter is fine-tuned and applied to each test subset individually. For instance, $\mathcal{M} + \mathcal{Q}_{\text{MATH}}$ is fine-tuned on the MATH training subset of MATH-Combined and evaluated on its corresponding MATH test subsets; similarly, $\mathcal{M} + \mathcal{Q}_{\text{GSM8K}}$ is fine-tuned on the GSM8K training subset and evaluated on the GSM8K test subsets, and so on. dd We also include the **backbone model** $\mathcal{M}$ itself as a baseline, which is used directly for test-case inference without any adapter fine-tuning. This baseline is applied only to BBH and MMLU datasets, as they contain in-context examples to guide the model's output format.

All other baseline methods start from the $\mathcal{M} + \mathcal{Q}_{\text{base}}$ checkpoint for further fine-tuning or inference, and include:

- **MoE Routing**: This baseline implements layer-level routing with the same weights as ELREA. Specifically, similar to Equation 5.3, the averaged linear layer adapter output is given by

$$\mathcal{F}(\boldsymbol{x}) = \sum_{c=0}^{C} \lambda_c \boldsymbol{B}_c \boldsymbol{A}_c^{\mathsf{T}} \boldsymbol{x}; \quad \lambda_c = \frac{w_c}{\sum_{c'=0}^{C} w_{c'}}; \quad w_0 \triangleq w_{\text{base}}, \boldsymbol{A}_0 \triangleq \boldsymbol{A}_{\text{base}}, \boldsymbol{B}_0 \triangleq \boldsymbol{B}_{\text{base}}. \tag{D.1}$$

  Here, we omit the layer indicator i for simplicity. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined as in subsection 5.2.1, and w represents the routing weight for each cluster as in Equation 5.9. Note that $\mathcal{F}(\boldsymbol{x})$ is the output of the LoRA MoE, which should be added to the layer output from the backbone model $\mathcal{M}(\boldsymbol{x})$ with a scaling factor of $\alpha/r = 4$, as mentioned in Appendix section D.2.

- **MoE Merging**: This baseline merges the expert network weights before processing the input. Specifically, the averaged linear layer adapter weights become the final weights for the model, *i.e.*, $\boldsymbol{A} = \sum_{c=0}^{C} \lambda_c \boldsymbol{A}_c$ and $\boldsymbol{B} = \sum_{c=0}^{C} \lambda_c \boldsymbol{B}_c$. Once merged, the network behaves as a single-expert model, and the output is calculated as $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{x}$.

- **Mixture of LoRA Experts** (**MoLE**, [192]): This baseline models each layer of trained LoRAs as a distinct expert and incorporates a learnable gating function within each layer, in contrast to the precomputed universal routing weights used in MoE Routing. Using the same notation as in Equation D.1, the output of each MoLE layer is defined as

$$\mathcal{F}(\boldsymbol{x}) = \sum_{c=0}^{C} \lambda_c \boldsymbol{B}_c \boldsymbol{A}_c^{\mathsf{T}} \boldsymbol{x}; \quad \lambda_c = \frac{\exp(\boldsymbol{w}_c^{\mathsf{T}} \boldsymbol{x})}{\sum_{c'=0}^{C} \exp(\boldsymbol{w}_{c'}^{\mathsf{T}} \boldsymbol{x})}, \tag{D.2}$$

  where $\boldsymbol{w}_c$, a vector of the same dimensionality as $\boldsymbol{x}$, represents the learnable gating weight of a single-output linear layer for each expert c. In our setup, the gating outputs are expected to exhibit an imbalanced distribution, as shown in Figure 5.2.

Consequently, we do not include the gating balancing loss proposed by [192]. The routing parameters are trained on the entire training set for 1 epoch at a learning rate of 2×10^{-5} with all other parameters frozen.

- **LoRA Ensembles** [177]: This baseline trains three adapters, $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$, independently on the entire dataset using the same configuration as the base adapter $\mathcal{Q}_{\text{base}}$ (subsection 5.3.1). During inference, four models (*i.e.*, $\{\mathcal{M} + \mathcal{Q}^{(e)}_{\text{base}}\}$ and $\{\mathcal{M} + \mathcal{Q}_i\}_{i=1}^{3}$) are applied to the input sequence. The final prediction is then computed by averaging their pre-activation logits and taking the ArgMax as the predicted next token. We do not match the number of ensemble models to the number of clusters, C, in ELREA due to concerns about the training and evaluation costs.
- **Self-Consistency** [132]: This baseline performs 5 separate inference passes with $\mathcal{M}+\mathcal{Q}_{\text{base}}$ for each instance, using random token sampling with the last-layer SoftMax activation temperature set to 1. The final answer is determined by majority voting among the 5 predictions. In case of a tie, one of the tied answers is randomly selected as the final prediction.
- **Instruction Embedding**: Instead of using the instruction gradients representation from Equation 5.2, this baseline employs the sentence embedding of the instruction text directly for training data clustering and test instance routing. Specifically, we use the Sentence Transformers [126] Python package with the `all-mpnet-base-v2` model checkpoint[3] to encode the instruction text into a fixed-size vector, which is then used for clustering and routing in the same way as the gradient features.
- **Random Cluster**: This baseline maintains the same number of clusters and cluster sizes as ELREA but assigns cluster members randomly from the fine-tuning dataset $\mathcal{D}_{\text{ft}}$. Specifically, $\mathcal{D}_{\text{rand},c} \subset \mathcal{D}_{\text{ft}}$, with $|\mathcal{D}_{\text{rand},c}| = |\mathcal{D}_c|$, and $\mathcal{D}_{\text{rand},c} \cap \mathcal{D}_{\text{rand},c'} = \varnothing$ for all $c \neq c' \in \{1, 2, \ldots, C\}$. The corresponding adapters are fine-tuned on these randomly assigned clusters and are uniformly weighted during inference, *i.e.*, $w_{\text{rand,base}} =$

[3] https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

$w_{\text{rand},1} = \ldots = w_{\text{rand},C} = 1$. This random assignment preserves the distribution characteristics of $\mathcal{D}_{\text{ft}}$, positioning Random Cluster as an approximate deep ensemble baseline with equivalent training effort to ELREA.

- **Uniform Weights**: This baseline assigns uniform weights to all clusters during inference, *i.e.*, $w_{\text{base}} = w_1 = \ldots = w_C = 1$.

D.5 Efficiency Analysis

Theoritical Analysis Theoretically, the computational overhead of ELREA compared to using $\mathcal{M} + \mathcal{Q}_{\text{base}}$ arises from the following aspects:

1) the computation of the gradients of all training and test instructions; 2) clustering the gradient features of the training data points and computing the weights of each test data point on the clusters; 3) additional training steps to fit LoRA experts on the training clusters; 4) additional computational resources required to perform the forward pass on all LoRA experts for each test data point. In practice, step 2) only takes a few minutes with our clustering setup (subsection 5.3.3 and subsection 5.3.4), which is negligible compared to the entire training process and will be ignored in the following discussion.

If implemented properly, step 1) can also be integrated into the training and inference process with relatively small overhead. With a naïve implementation, step 1) approximately equals the cost of training the model on the combination of training and test *instructions* (without answers) for one epoch, whose overhead depends on the average length of the instructions. For datasets such as OpenAssistant, MATH, GSM8k, and MathQA, whose average instruction length is comparatively much shorter than the answer length (Table 5.1), the overhead is minimal. In the worst-case scenario, step 1)'s overhead approximates the cost of training the model on the combination of training and test for one epoch, which is still acceptable for most fine-tuning datasets.

As the sum of our training cluster sizes equals the number of training data points, *i.e.*,

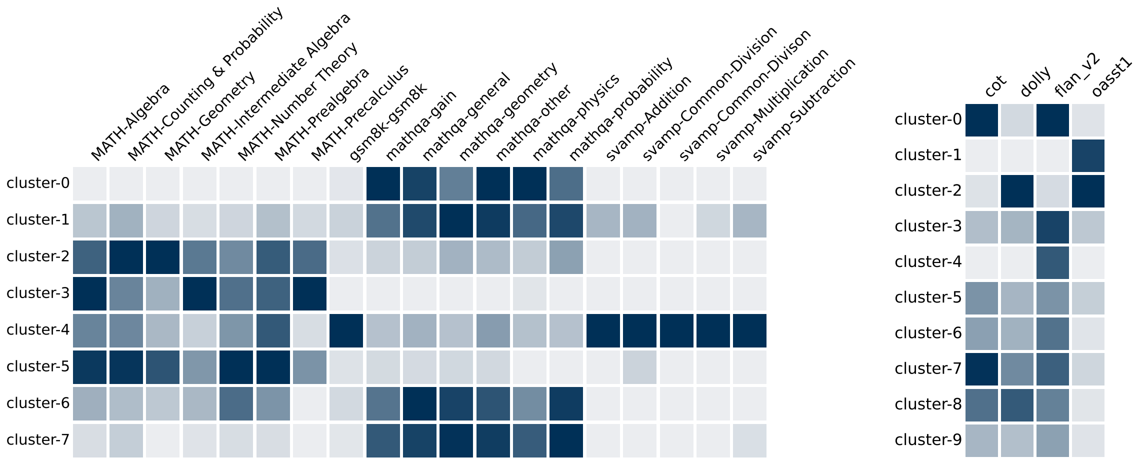
Table D.1: Efficiency comparison on a toy dataset. Time is in seconds; memory is in GiB.

Step	$\mathcal{M} + \mathcal{Q}_{\text{base}}$		ELREA	
	Time	Memory	Time	Memory
Fine-tuning base adapter $\mathcal{Q}_{\text{base}}$ on $\mathcal{D}_{\text{ft}}$ (subsection 5.3.1)	246	15.49	246	15.49
Calculating training gradient features $\boldsymbol{\delta}(\boldsymbol{x}_{\text{ft, instr}})$ (subsection 5.3.3)	–	–	68	24.76
Calculating test gradient features $\boldsymbol{\delta}_{\text{test}}$ (subsection 5.3.4)	–	–	14	24.76
Fine-tuning experts on clusters (subsection 5.3.3)	–	–	246	15.49
Fine-Tuning Total	246	–	574	–
Inference (subsection 5.3.4)	114	7.73	262	18.46

$\sum_{c=1}^{C} |\mathcal{D}_c| = |\mathcal{D}_{\text{ft}}|$, the additional training steps in step 3) take the same amount of time as training the base adapter $\mathcal{Q}_{\text{base}}$ (subsection 5.3.4) on $\mathcal{D}_{\text{ft}}$, excluding CPU-disk I/O overhead, which is generally less than one minute in our experiments.

The complexity of step 4), however, is harder to estimate as it varies drastically according to the implementation. In our implementation, we choose to duplicate the input instruction along the batch dimension by the number of experts (*i.e.*, $C + 1$) and perform a forward pass on the backbone and all experts simultaneously. This implementation has a similar cost to using a $(C + 1)\times$ inference batch size with the base adapter $\mathcal{M} + \mathcal{Q}_{\text{base}}$.

Empirical Results To evaluate the efficiency of ELREA, we compared its computation time with that of the baseline model $\mathcal{M} + \mathcal{Q}_{\text{base}}$ using a same set of hyper-parameters and device configuration on a single NVIDIA A101 80G GPU, except for the following specific parameters. We generate a **toy** dataset consisting of 2,000 training samples and 400 test samples as a smaller-scale but more controllable evaluation setup. Each sample contains 60 random *lorem-ipsum* words in both the instruction and the answer (which accounts for around 200 tokens each), matching the lengths in Dolly-15k (Table 5.1). We designate $C = 4$ experts and set the LoRA ranks to $r = 8$. The model undergoes fine-tuning over 3 epochs, with batch sizes of 4 for both fine-tuning and inference. During inference, the model consistently predict the next 20 tokens for all input instructions to ensure a fair comparison.



(a) MATH-Combined

(b) GLUR

Figure D.1: Distribution of data sources and categories within each cluster for the MATH-Combined and GLUR (general language understanding and reasoning) training sets at rank $r = 8$. Cluster indices are shown along the rows, while columns represent data sources and categories, formatted as "{`source dataset`}-{`category`}" for MATH-Combined and "{`source dataset`}" for GLUR. The color intensity reflects the sample count, with darker shades indicating higher counts. Each column is independently normalized, meaning scales may differ across columns. Color gradients are slightly curved to improve visibility for categories with fewer samples.

The results from our implementation, presented in Table D.1, indicate that the fine-tuning time for ELREA was 574 seconds, which is approximately $2.3\times$ that of the baseline $\mathcal{M}+\mathcal{Q}_{\text{base}}$'s 246 seconds. Similarly, the inference time and memory consumption are about $2.3\times$ and $2.4\times$, respectively. In contrast, a classic Deep Ensembles setup, where each LoRA expert is trained independently from scratch on the entire dataset, would require $5\times$ the time of the baseline for both fine-tuning and inference. Thus, ELREA offers significant efficiency and performance gains compared to this more traditional approach.

Further enhancements to ELREA' efficiency could be achieved by reducing the number of experts or the LoRA ranks, or by constructing gradient features from only the top-k Transformer blocks rather than the entire model. Moreover, we are exploring LoRA merging techniques in ongoing work to effectively combine similar expert adapters, thereby further reducing inference costs.

D.6 Further Analysis on Data Clustering

To better understand the distribution of data across clusters, we analyzed the sources and categories within each cluster from the MATH-Combined dataset, as visualized in Figure D.1. Here, “data source” refers to the individual datasets that comprise MATH-Combined (*i.e.*, MATH, GSM8k, SVAMP, or MathQA) and language understanding and reasoning (*i.e.*, CoT, Dolly-15k, Flan V2, and OpenAssistant), and “category” pertains to the finer-grain labels within these datasets. Notably, GSM8k is categorized uniformly under a single label “`gsm8k`” due to its lack of distinct category labels.

Analysis of Figure D.1 reveals distinct correlations between clusters and data sources. For instance, in MATH-Combined, clusters 2, 3, and 5 predominantly contain samples from MATH, whereas clusters 0, 1, 6, and 7 primarily feature contributions from MathQA. This clustering also appears to group together tasks requiring similar mathematical skills; for example, cluster 4 heavily includes SVAMP samples, which typically assess algebraic problem-solving capabilities, alongside significant portions of “Algebra” and “Pre-Algebra” from the MATH dataset.

Additionally, within individual sources, clusters distinguish between finer categories effectively; cluster 2 mainly focuses on Geometry and Probability, whereas cluster 3 is concentrated on Algebra. These insights suggest that the data representations successfully capture inherent structural differences, making the clustering both interpretable and meaningful. Such characteristics motivates the design of ELREA and significantly improves its efficacy.