Squeezing Your Fine-Tuning Data to the Last Drop
From Selection to Rebalancing

Minghao Wu

Department of Data Science and Artificial Intelligence
Monash University

November 14, 2025

About Me

- Final-year Ph.D. candidate at Monash University
- Research focus: Large Language Models, Multilinguality, Machine Translation
- 20+ papers in top-tier venues (ICML, ACL, EMNLP, COLING, TACL, etc.)
- Outstanding Paper Award at ACL 2025
- Visit/Intern experience: Huawei, Tencent, Alibaba, MBZUAI
- https://minghao-wu.github.io/

# Overview

The Data Challenge in LLM Fine-Tuning
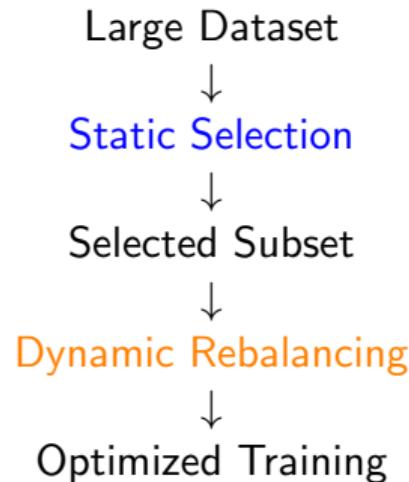
Why does data matter in LLM fine-tuning?

- Training data directly shapes model capabilities
- Quality determines how well the model learns
- Diversity ensures comprehensive skill coverage
- Optimization affects learning effectiveness

Key Challenges:

- Selection: What data to train on?
- Composition: How to mix datasets?
- Optimization: When to use what data?

The Pipeline

Large Dataset
↓
Static Selection
↓
Selected Subset
↓
Dynamic Rebalancing
↓
Optimized Training

*Two complementary approaches*

Two Complementary Approaches
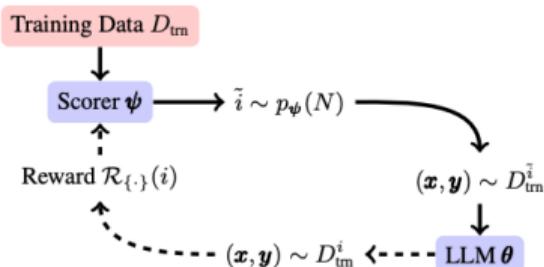
Study 1: GraphFilter

- Static Data Selection
- Bipartite graph representation
- Set cover optimization
- Quality × Diversity priority

Study 2: Mixture-of-Skills

- Dynamic Rebalancing
- Reinforcement learning framework
- Adaptive data rebalancing
- Skill-aware training
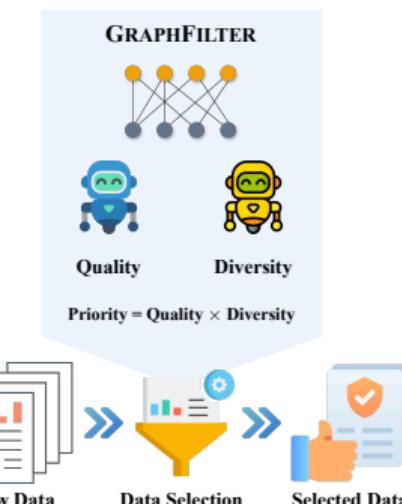
The Best of Both Worlds: Bridging Quality and Diversity in Data Selection with Bipartite Graph

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, Gholamreza Haffari

Key Contributions:

- Novel formulation as set cover problem
- Bipartite graph representation
- Multiplicative priority function
- Outperforms 9 baselines on 6 benchmarks

The Quality-Diversity Dilemma

Existing Methods Fall Short:

- Quality-focused: Select high-scoring examples
- Diversity-focused: Maximize coverage
- Problem: One aspect sacrificed for the other

Real-world Analogy:

- Curating a library collection
- Want both high-quality books *and* diverse topics
- Balance is key for comprehensive learning

The Challenge

Quality ↑
↓
Diversity ↓

GraphFilter: Best of Both Worlds

What is the Set Cover Problem?

Classic Computer Science Problem:

- Given a universe of elements U
- Given a collection of sets $\mathcal{S} = \{S_1, S_2, ..., S_m\}$
- Each set $S_i \subseteq U$
- **Goal**: Find minimum number of sets that cover all elements in U

NP-Hard Problem:

- No polynomial-time exact solution
- Greedy approximation works well
- Widely applicable in real-world scenarios

Mathematical Formulation

$$\min \sum_{i=1}^{m} x_i$$
subject to:
$$\sum_{i:e \in S_i} x_i \geq 1, \forall e \in U$$
$$x_i \in \{0, 1\}$$

where $x_i = 1$ if set S_i is selected

From Set Cover to Data Selection

The Connection:

- Universe U: All possible n-grams
- Sets $\mathcal{S}$: Training sentences
- Coverage: Each sentence covers its n-grams
- Goal: Select sentences that cover diverse n-grams

Why This Makes Sense:

- N-grams represent linguistic patterns
- Diverse n-gram coverage = diverse training data
- Natural formulation for diversity

Data Selection as Set Cover

Sentence 1: "How to cook pasta"

**N-grams**: {how, to, cook, pasta, how_to, to_cook, ...}
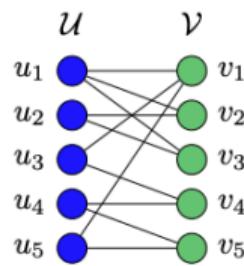
Sentence 2: "How to bake bread"

N-grams: {how, to, bake, bread, how_to, to_bake, ...}
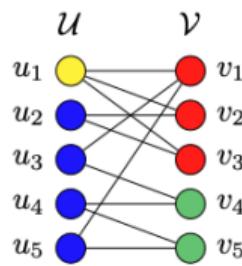
Bipartite Graph Representation

Graph Components:

- Left nodes: Sentences $u_1, u_2, ..., u_5$
- Right nodes: N-grams $v_1, v_2, ..., v_5$
- Edges: Connect sentences to their n-grams
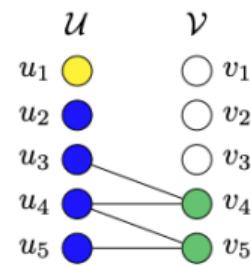
Selection Process:

- Select sentence with highest priority
- Remove covered n-grams from graph
- Update priorities dynamically
- Repeat until budget exhausted

(a) Initial Graph   (b) Selection   (c) Remove Edges   (d) Remove Nodes

Figure: Example: Selecting u_1 covers n-grams $\{v_1, v_2, v_3\}$, then selecting u_4 covers remaining n-grams

But wait... This only considers diversity, not quality!

Priority Function Design:

$$\phi(u) = \text{Quality}(u) \times \max(1, \text{Diversity}(u))$$

Quality Metric - IFD Score:

$$\text{Quality}(u) = \text{PPL}(y|x)/\text{PPL}(y)$$

Diversity Metric - TF-IDF:

$$\text{Diversity}(u) = \sum_{v \in \mathcal{V}_u} \text{TF-IDF}(v)$$

Why Multiplicative?

High Quality + Low Diversity
$\Rightarrow$ Low Priority

Low Quality + High Diversity
$\Rightarrow$ Low Priority

High Quality + High Diversity
$\Rightarrow$ **High Priority**

Both aspects must be strong for selection!

GraphFilter Algorithm in Action

Iterative Selection Process:

1. **Initialize**: Empty selection $\mathcal{S} = \emptyset$
2. **Compute**: Priority $\phi(u)$
3. **Select**: u^* with highest priority
4. **Update**: Remove u^* and n-grams
5. **Repeat**: Until budget k is reached

Dynamic Priority Updates:

- Priorities change as n-grams are covered
- Encourages selection of complementary sentences
- Balances quality and remaining diversity

Computational Efficiency

- Max-heap $\mathcal{O}(\log N)$ per iteration
- Localized priority updates
- Scalable to large datasets

N-gram Selection

- Use n-grams up to length 3
- Balances granularity and efficiency
- Captures meaningful linguistic patterns
- Empirically effective

Experimental Setup

Training Dataset:
- **Magpie** dataset (300K instances)
- High-quality instruction-response pairs
- Select 10K subset

Model Backbones:
- Gemma-2-2B
- Mistral-7B-v0.3
- Llama-3-8B

Baseline Methods (9 total):
- **Heuristic**: Random, Longest
- **Quality-based**: PPL, ArmoRM, AlpaGasus, DEITA, IFD
- **Diversity-based**: K-means, InsTag

Evaluation Benchmarks (6 total):
- MMLU, ARC, HellaSwag, GSM8K
- AlpacaEval-2.0, MT-Bench using GPT-4o

Consistent Improvements Against Random:

- **Gemma-2-2B**: Up to $+2.03$
- **Mistral-7B**: Up to $+2.83$
- **Llama-3-8B**: Up to $+2.46$

Key Findings:

- GraphFilter achieves best/second-best on most benchmarks

- Quality-only methods show benchmark bias (e.g., ArmoRM good on AlpacaEval, poor elsewhere)

	Standard	LLM	ALL
Random	47.75	41.04	45.51
Longest	46.91	39.96	44.59
Perplexity	48.27	40.28	45.61
ArmoRM	48.21	<u>42.66</u>	46.36
AlpaGasus	48.96	41.90	46.60
DEITA	48.78	41.70	46.42
SuperFilter	49.10	41.91	46.70
Kmeans	48.90	41.72	46.51
InsTag	<u>49.93</u>	41.72	<u>47.19</u>
GraphFilter (Ours)	**50.55**	**42.79**	**47.97**

Table: Main results given by Llama-3-8B on the standardized benchmarks and LLM-as-a-Judge benchmarks. The best results are highlighted in **bold**, and the second-best results are highlighted in <u>underline</u>.

Main Results: GraphFilter is Computationally Efficient

Computational Efficiency

- Max-heap $\mathcal{O}(\log N)$ per iteration
- Localized priority updates
- Scalable to large datasets

	Runtime (hrs)
Perplexity	0.92
ArmoRM	5.93
AlpaGasus	32.34
DEITA	22.65
SuperFilter	1.95
Kmeans	2.26
InsTag	25.48
GraphFilter (Ours)	2.48
w/o priority $\phi(u)$	$0.53^{\dagger}$

Table: Runtime (in hours) for selecting 10K training instances. $\dagger$ indicate the CPU-only method.

Visualization Analysis:

- **Lexical Diversity**: Measured by MTLD metric
- **Data Quality**: Assessed by SkyworkRM
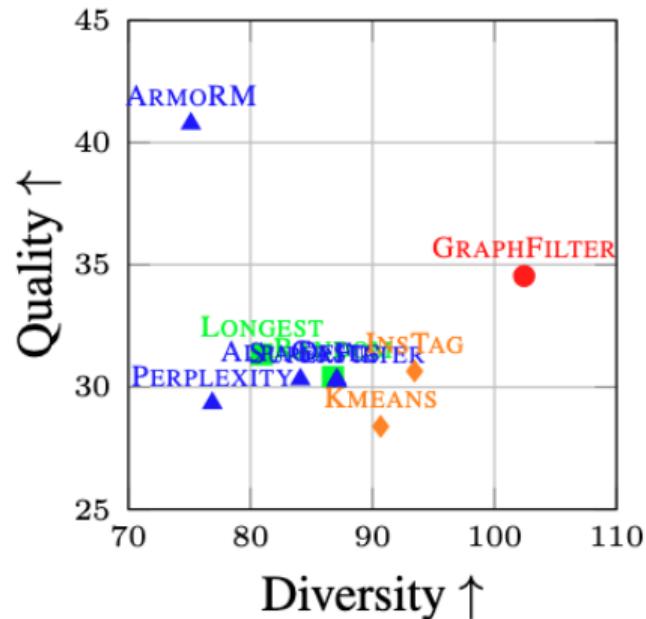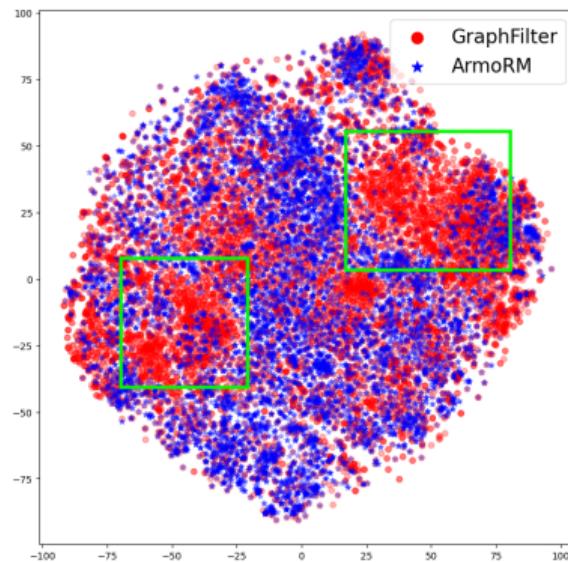
Figure: GraphFilter achieves highest lexical diversity and second-best data quality among all methods.
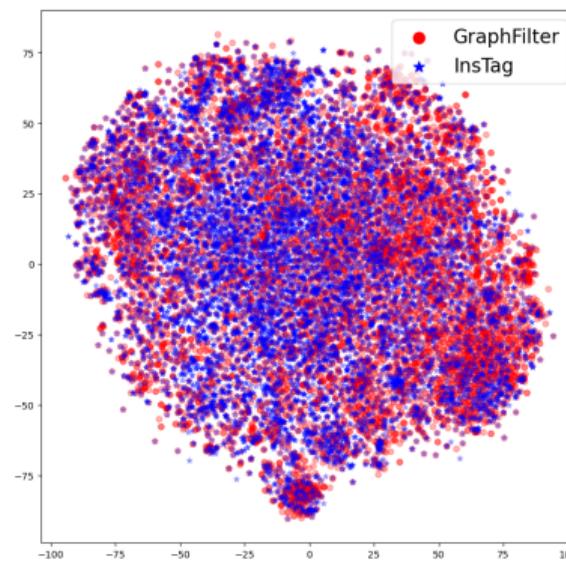
**Semantic Visualization**: t-SNE plots using BGE embeddings

- Selects instructions not chosen by quality-only methods
- Similar semantic diversity to diversity-focused methods

(a) GraphFilter vs. ArmoRM (Quality only)   (b) GraphFilter vs. InsTag (Diversity only)

Analysis: N-gram Combinations Capture Multi-Level Features

Different N-grams Capture Different Patterns:

- **Unigrams (n=1)**: Individual words, basic vocabulary
- **Bigrams (n=2)**: Local word relationships
- **Trigrams (n=3)**: Phrasal patterns, syntax

Key Finding:

- Combining 1-grams + 2-grams + 3-grams significantly outperforms individual n-gram types
- Each level provides complementary information
- Integration consolidates features effectively

Performance by N-gram Type

N-gram	ALL
1-gram only	46.48
2-gram only	46.63
3-gram only	47.15
1+2+3-grams	**47.97**

Multi-level features are essential!

Analysis: Trigrams Provide Optimal Balance

N-gram Size Analysis (n_{max}=1 to 5):

- **Performance**: Peaks at trigrams (n_{max}=3)
- **Efficiency**: Decreases with larger n_{max}
- **Diminishing returns**: Beyond n_{max}=3

Why Trigrams Work Best:

- Capture meaningful linguistic patterns
- Balance granularity and computational cost
- Avoid over-specification of larger n-grams

N-gram Size Trade-offs

n_{max}	ALL	Time (hrs)
1	46.48	2.12
2	47.31	2.30
3	**47.97**	**2.48**
4	47.43	3.38
5	47.85	4.58

Sweet spot at n=3

\# N-grams Nodes:
0.1M (1-gram) $\rightarrow$ 7.4M (5-gram)

What to Apply GraphFilter To?

- Each training instance has instruction + response

- Tested three scenarios:
  - Instructions only
  - Responses only
  - Both instructions and responses

Surprising Result:

- **Instructions only performs best**

- Instruction diversity more impactful than response diversity

- Quality remains similar

- Rethinking: Response diversity also matters!

Type	Benchmarks			Lexical Diversity		Quality
	Standard	LLM	ALL	Instruction	Response	
Instruction	**50.55**	**42.79**	**47.97**	**102.43**	71.74	**81.54**
Response	47.16	39.71	44.68	90.22	**73.57**	81.52
Inst. + Resp.	48.03	41.20	45.76	90.13	72.60	81.52

When to Prioritize What?

- **Small budgets (1K, 5K)**: Quality methods (SuperFilter) excel

- **Large budgets (10K+)**: Diversity methods (InsTag) catch up

- **All budgets**: GraphFilter consistently demonstrates performance gains
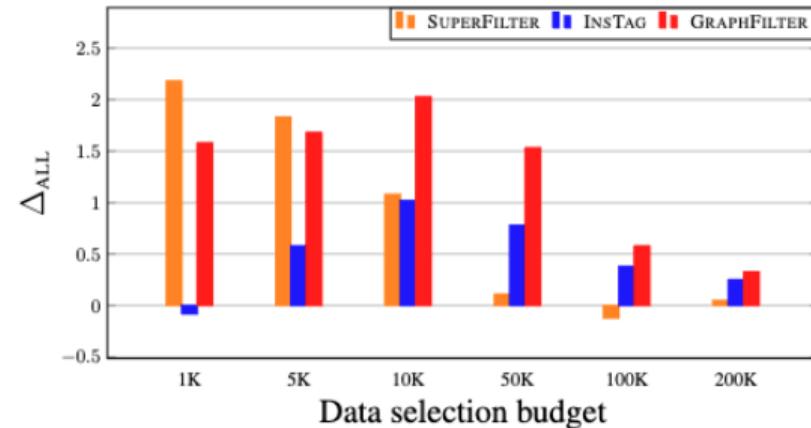
Figure: Relative gains against Random. Budget size affects strategy effectiveness

GraphFilter Insights:
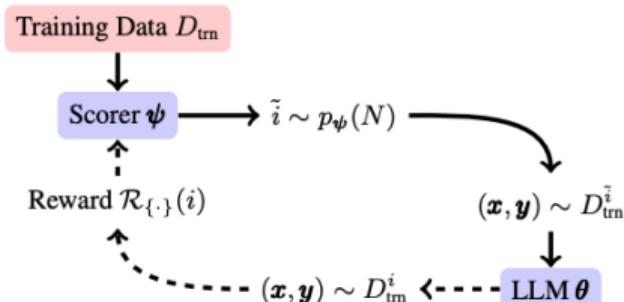
- Quality $\times$ Diversity balance is crucial
- Static selection has limitations
- What about during training?

New Challenge:

- Models learn different skills at different rates
- Static data mixing may not be optimal
- Finding the right data mix is expensive

The Next Question

Can we optimize data usage dynamically during fine-tuning?

Enter: Mixture-of-Skills
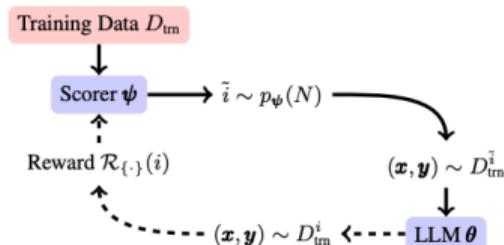
Mixture-of-Skills: Learning to Optimize Data Usage for Fine-Tuning Large Language Models

Minghao Wu, Thuy-Trang Vu, Lizhen Qu, Gholamreza Haffari

2024 Conference on Empirical Methods in Natural Language Processing

Key Contributions:

- Reinforcement learning framework for data rebalancing
- Model-agnostic dynamic optimization
- Adaptive skill development during training
- MoSpec extension for task-specific fine-tuning

The Data Mixing Challenge

LLMs Need Multiple Skills:

- Writing, coding, mathematics, chatting, etc.
- Each skill requires different training data
- Datasets are heterogeneous and imbalanced

Current Problems:

- **Static mixing**: Ignores learning dynamics
- **Data capping**: Limits large dataset utilization
- **Grid search**: Finding optimal composition is expensive

Example Challenge

- Math: 10K samples
- Code: 100K samples
- Chat: 1M samples

Learning Dynamics:

- Some skills learned quickly
- Others need more exposure
- Skills can interfere

Mixture-of-Skills: Bilevel Optimization

The Framework:

- Outer level: Optimize LLM parameters $\boldsymbol{\theta}$
- Inner level: Optimize data sampling via scorer network $\boldsymbol{\psi}$
- **Goal**: Learn optimal data usage automatically

Mathematical Formulation:

$$\boldsymbol{\psi} = \underset{\boldsymbol{\psi}}{\arg\min} \, \mathcal{J}(D_{\mathrm{trn}}; \boldsymbol{\theta}(\boldsymbol{\psi}))$$

$$\boldsymbol{\theta}(\boldsymbol{\psi}) = \underset{\boldsymbol{\theta}}{\arg\min} \, \mathbb{E}_{i \sim P_{\boldsymbol{\psi}}(N)}[\mathcal{L}(D_{\mathrm{trn}}^{i}; \boldsymbol{\theta})]$$

Why RL?

- Bilevel optimization not directly differentiable
- REINFORCE algorithm handles discrete sampling
- Enables dynamic adaptation

MoS Algorithm: Step-by-Step Walkthrough

Algorithm Overview:

1. **Initialize**: Start with uniform sampling $(\tau = \infty)$
2. **Training Loop**: For each step t:
   - Sample dataset $\tilde{i} \sim p_{\boldsymbol{\psi}}(N)$
   - Sample batch from $D_{\mathrm{trn}}^{\tilde{i}}$
   - Update LLM: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}$
3. **Scorer Update** (every S steps):
   - Sample batch from each dataset
   - Compute rewards $\mathcal{R}(i)$ for each dataset
   - Update scorer:
     $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \gamma \sum_i \mathcal{R}(i) \nabla_{\boldsymbol{\psi}} \log p_{\boldsymbol{\psi}}(i)$

Key Components:

- **Scorer Network ψ**: Simple 2-layer MLP
- **Update Frequency S**: Computational efficiency
- **Rewards $\mathcal{R}(i)$**: Guide optimization

Dynamic Adaptation:

- Sampling probabilities change over time
- Responds to current model state
- Balances different skills automatically

But how do we design rewards?

Reward Design: Three Perspectives

1. **Transferability ($\mathcal{R}_{\mathsf{cosine}}$):**
   - Measure similarity between datasets
   - Use mini-batch embeddings from LLM
   - Higher similarity = better transferability

   $$\mathcal{R}_{\mathsf{cosine}}(i) = \frac{1}{N} \sum_{n=1}^{N} \frac{\mathbf{z}^i \cdot \mathbf{z}^n}{\|\mathbf{z}^i\| \cdot \|\mathbf{z}^n\|}$$

2. **Difficulty ($\mathcal{R}_{\mathsf{diff}}$):**
   - Relative perplexity decrease after fine-tuning
   - Higher values = more difficult datasets need more attention

   $$\mathcal{R}_{\mathsf{diff}}(i) = \frac{1}{L} \sum_{j=1}^{L} \frac{\mathrm{PPL}(\mathbf{y}_j^i; \mathbf{x}_j^i, \boldsymbol{\theta})}{\mathrm{PPL}(\mathbf{y}_j^i; \mathbf{x}_j^i, \boldsymbol{\theta}_0)}$$

3. **Learning Trajectory:**
   - Exponential Moving Average (EMA) for stability
   - Smooths reward signals over time
   - Prevents oscillations

   $$\mathcal{R}(i) = \beta \mathcal{R}'(i) + (1 - \beta)\mathcal{R}''(i)$$

Design Principles:
   - **Transferability**: Knowledge sharing
   - **Difficulty**: Learning progress
   - **Trajectory**: Stability

Rewards guide the scorer network to make informed sampling decisions

Experimental Setup

Datasets (4 Skills):

- **Mathematics**: MathInstruct (comprehensive collection)
- **Medicine**: MedInstruct (medical instructions)
- **General**: ShareGPT (diverse, high-quality conversations)
- **NLP**: P3 (660 subsets, diverse NLP tasks)

Model Backbones (3 total):

- Qwen1.5-0.5B
- Gemma-2B
- Llama-3-8B

Baselines:

- **Heuristic**: Temperature sampling

$$q_\tau(i) = \frac{q(i)^{1/\tau}}{\sum_{n=1}^{N} q(n)^{1/\tau}}$$

  - Proportional ($\tau = 1$), Temperature ($\tau = 10$), Uniform ($\tau = \infty$)
- **Dynamic**: MultiDDS, MultiUAT

Evaluation:

- **MMLU**: 57 subjects (Math, Medicine, Others)
- **MT-Bench**: 8 skills (Coding, Writing, etc.)

Main Results: MoS Consistently Outperforms Baselines

Consistent Improvements:

- **Qwen1.5-0.5B**: $+0.96$ improvement
- **Gemma-2B**: $+1.15$ improvement
- **Llama-3-8B**: $+2.45$ improvement

Key Findings:

- Larger models benefit more from MoS
- No universally optimal temperature τ
- Different rewards work better for different models
- EMA consistently helps performance

	MMLU	MT	ALL
Prop. ($\tau = 1$)	56.78	6.52	60.97
Temp. ($\tau = 10$)	56.17	6.66	61.40
Uni. ($\tau = \infty$)	55.72	6.63	60.99
MultiDDS	56.65	6.69	61.77
MultiUAT	55.66	6.67	61.18
MoS + cos	56.95	6.80	62.49
MoS + cos + EMA	<u>58.08</u>	**6.96**	**63.85**
MoS + diff	57.93	6.81	63.00
MoS + diff + EMA	**58.34**	<u>6.82</u>	<u>63.26</u>

Table: Results on MMLU and MT-Bench given by Llama-3-8B. Best results in **bold**, second-best <u>underlined</u>.

Dynamic Sampling Behavior:

- Sampling probabilities evolve over training
- Reflects changing model needs
- Balances skill development

(a) MOS + COSSIM + EMA   (b) MOS + DIFF + EMA
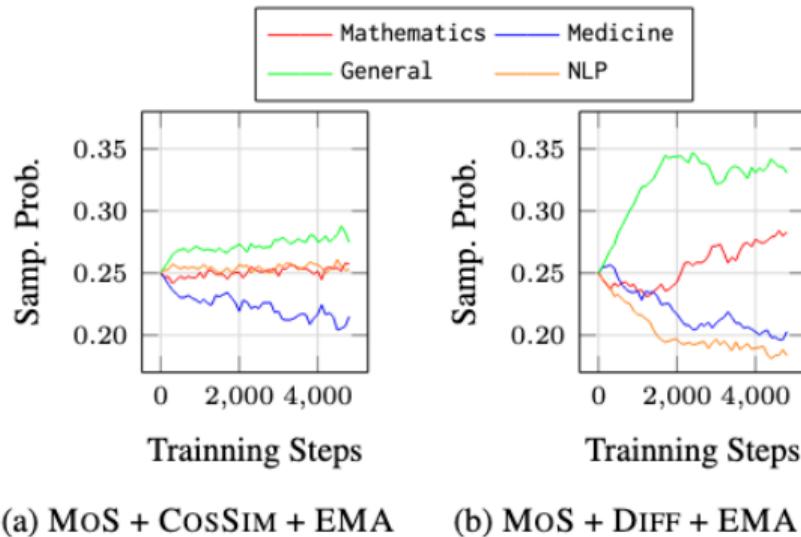
Figure: Sampling probabilities for each dataset over training steps. MoS adjusts focus dynamically.

Initialization Sensitivity:

- MoS starts with uniform sampling ($\tau = \infty$)

- What if we start with different priors?

- **Key Finding**: MoS consistently outperforms best heuristic baseline regardless of initialization

Robustness Analysis:

- All MoS variants beat Temperature ($\tau = 10$) baseline

- Proper prior selection can further enhance performance

- **MoS is robust and effective**

	Prior τ	MMLU	MT	ALL
Heuristic	10	56.17	6.66	61.40
MultiDDS	1	56.65	6.69	61.77
	10	56.88	6.58	61.34
	∞	55.66	6.67	61.18
MoS	1	56.51	6.81	62.29
	10	58.22	**6.91**	**63.66**
	∞	**58.34**	6.82	63.26

Table: MoS performance with different sampling priors using Llama-3-8B. All variants outperform the best heuristic baseline.

Takeaway: MoS learns to adapt regardless of how you start!

Analysis: MoS is Compatible with Data Selection Methods

Combining MoS with Instance Selection:

- Use Instruction-Following Difficulty (IFD) to select top 10% instances
- Apply MoS dynamic rebalancing on selected data
- **Best of both worlds**: Quality selection + dynamic optimization

Complementary Effects:

- Static selection improves data quality
- MoS optimizes usage of selected data
- **Complementary approaches** enhance final performance

	MMLU	MT	ALL
Random			
Temp. ($\tau = 10$)	54.89	6.63	60.62
MoS	55.21	6.72	61.15
IFD Selection			
Temp. ($\tau = 10$)	55.13	6.69	61.02
MoS	**56.43**	**6.77**	**62.05**

Table: Combining MoS with instance selection (IFD) further improves performance using Llama-3-8B.

Key Insight: Static selection and dynamic rebalancing address different aspects of data optimization!

MoSpec: Fine-tuning from Generalist to Specialist

The Specialization Challenge:

- Large general-purpose models are costly to deploy
- Many applications need only narrow functionalities
- Smaller specialized models often outperform larger generalist ones

MoSpec Approach:

- Harnesses diverse datasets to enhance target task performance
- Assigns higher rewards to target domain (e.g., mathematics)
- Learns optimal dataset distribution for specific capabilities

Example: Math Specialization

- Target: Mathematics dataset
- Supporting: General, Medicine, NLP datasets
- **MoSpec + cosine**: Compute similarity between Math and other datasets
- **MoSpec + diff**: Double reward for Math dataset

Key Insight: *SFT datasets from other domains are beneficial for the target task, especially when target dataset is incomplete*

MoSpec Results: Math Specialization Performance

Experimental Setup:

- **Target**: Math specialization
- **Baselines**:
  - Temp. ($\tau = 10$)
  - MathLlama (trained only on Math)
- **Evaluation**: GSM8K, MATH, MMLU-math (M-math) benchmarks

Key Findings:

- **MathLlama performs worst**
- **Other datasets are beneficial**
- **MoSpec learns optimal mixing** for specialization

	GSM8K	MATH	M-math	ALL
Generalist				
Temp. ($\tau = 10$)	49.62	9.54	28.36	29.17
MoS + cos + EMA	50.40	9.78	27.60	29.26
MoS + diff + EMA	49.58	10.26	**32.81**	30.88
Math Specialist				
MathLlama	41.02	9.76	30.34	27.04
MoSpec + cos + EMA	51.10	10.64	30.16	30.63
MoSpec + diff + EMA	**52.10**	**11.40**	32.24	**31.91**

Table: Math specialization results. MoSpec outperforms both generalist models and math-only training.

+4.87 improvement over math-only training!

Dynamic Data Usage:

- MoSpec adjusts dataset sampling over training
- Increases focus on target domain (Math)
- Leverages supporting datasets effectively
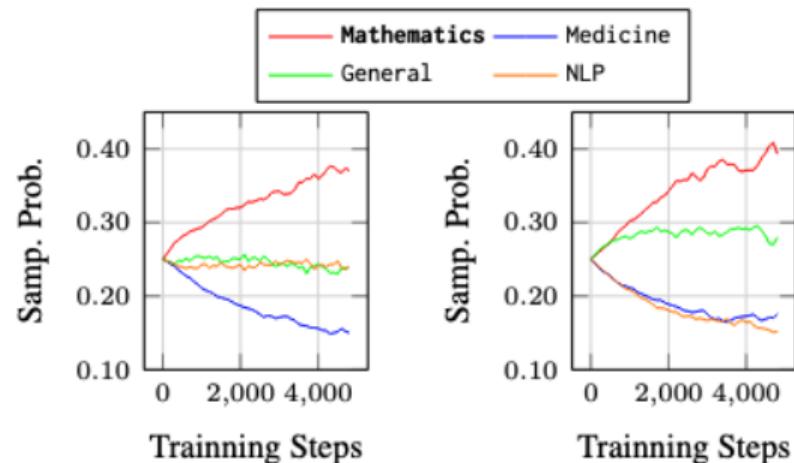
Figure: Sampling probabilities during Math specialization. **Left**: MoSpec + cos + EMA; **Right**: MoSpec + diff + EMA. Both increase focus on Math over time.

Summary of Contributions

GraphFilter: Principled Static Selection

- Novel set cover formulation for data selection
- Bipartite graph representation balancing quality $\times$ diversity
- Consistent improvements across models and benchmarks
- Computationally efficient and scalable

MoS: Intelligent Dynamic Optimization

- Model-agnostic RL framework for data rebalancing
- Adaptive skill development during fine-tuning
- Substantial performance improvements
- MoSpec extension for task specialization

Comprehensive solution covering the entire data optimization pipeline

Thank You

Questions & Discussion

Contact Information:

- Email: minghao.wu@monash.edu
- Twitter/X: https://x.com/WuMinghao_nlp
- LinkedIn: https://www.linkedin.com/in/minghao-wu-9087076a
- GitHub: https://github.com/minghao-wu
- Website: https://minghao-wu.github.io
- Papers: Available on arXiv and conference proceedings

Special thanks to ILCC for the invitation and the opportunity to share this work