Towards Trustworthy LLMs: Improving Robustness via Post-Training Optimization

Liang Chen

The Chinese University of Hong Kong

About Me

Liang Chen

- **Date of Birth:** October 1998 (Age: 26)
- **Ph.D. Candidate (3rd Year)**, Department of Systems Engineering and Engineering Management, CUHK
- **Advisor:** Prof. Kam-Fai Wong
- **Personal Website:** chanliang.github.io

Research Focus

LLM Trustworthiness

Robustness

Adversarial Attack
&
Distribution Shifts
[ICLR 2025]

Transparency

Long-CoT
&
Reasoning Model
[NIPS 2025 submission]

Validity

Factual Error
[EMNLP 2023]
Inconsistency
[ACL 2023 findings]

Resistance to Misuse

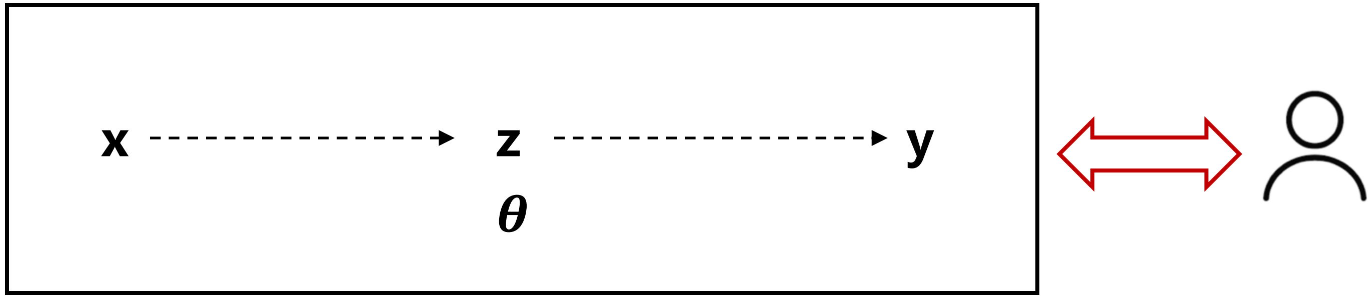
Cheating, Plagiarism
[ACL 2024]
Harmful Finetuning
[ICML 2025]

Today's Focus

Research Focus

LLM Trustworthiness

Robustness to Input	Transparency of Decision	Validity of Output	Resistance to Misuse
Adversarial Attack & Distribution Shifts [ICLR 2025]	Long-CoT & Reasoning Model [NIPS 2025 submission]	Factual Error [EMNLP 2023] Inconsistency [ACL 2023 findings]	Cheating, Plagiarism [ACL 2024] Harmful Finetuning [ICML 2025]



PEARL: TOWARDS PERMUTATION-RESILIENT LLMS

Liang Chen[1] **Li Shen**[2*] **Yang Deng**[3] **Xiaoyan Zhao**[1] **Bin Liang**[1] **Kam-Fai Wong**[1*]
[1]The Chinese University of Hong Kong [2]Shenzhen Campus of Sun Yat-sen University [3]SMU

Robustness in handling tasks with set-structure input, e.g. ICL, RAG

Backgroud: ICL Order Sensitivity

- **In-Context Learning (ICL) of LLMs: Powerful but Fragile**

  - Traverse all possible ordering, and calculate the average and worst-case performance.
  - Performance is highly sensitive to the order of demons.

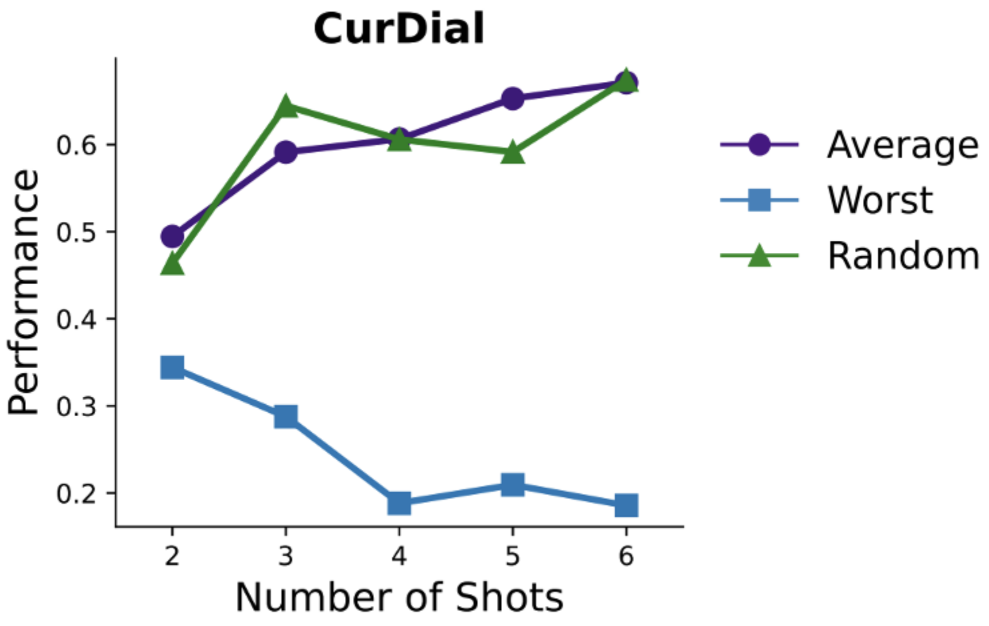


Figure 1: Performance of Llama-3 on CurDial datasets

Backgroud: ICL Order Sensitivity

- **The Fragility can be exploited to design a adversarial attack.**

  - ***Permutation Attack:*** attacker aims to fool LLMs by permuting ICL demonstrations.

  - ***Attack Success Rate:*** a sample is successfully attacked if its relative performance drop exceeds a threshold.

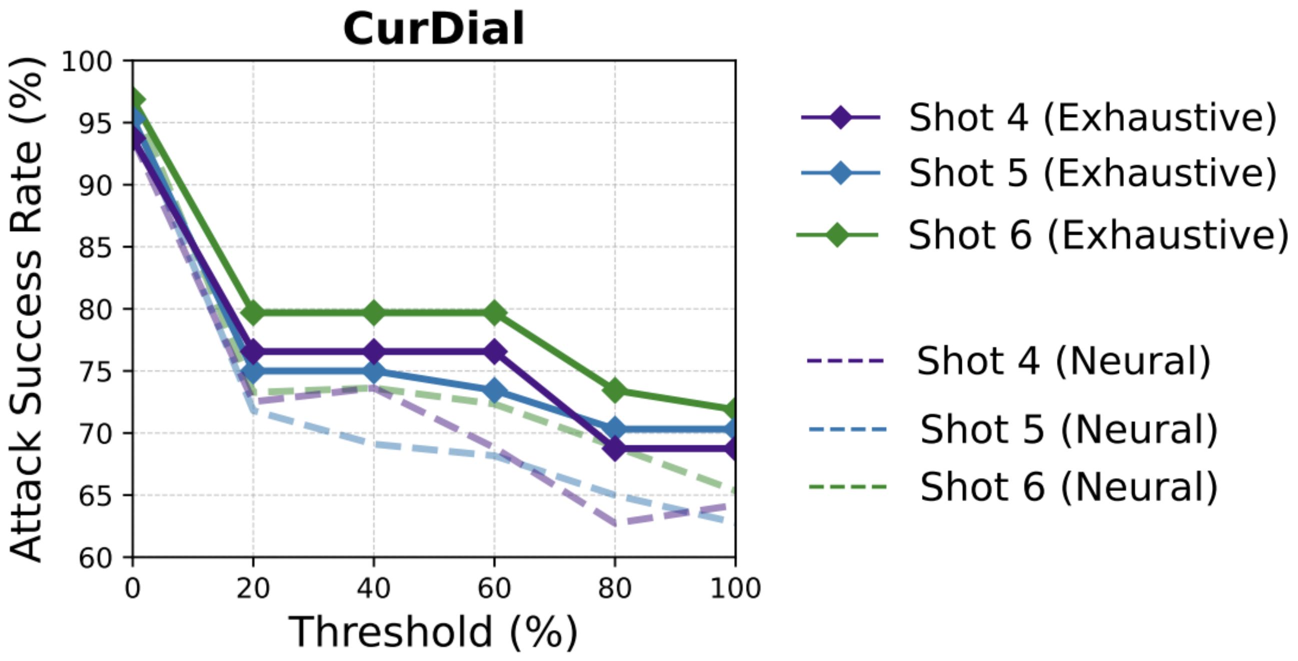


Figure 2: Attack success rates for exhaustive and neural search attack methods at different thresholds.

Backgroud: ICL Order Sensitivity

- **The Lack of robustness can**
  a) affect the user experience
  b) provide opportunities for malicious attacks

- **What are the *reasons* behind such non-robustness?**

Cause Analysis: Model Side

- **Autoregressive Nature of Transformer Architectural**
  - Positional encodings
  - Unidirectional attention

- **On the other hand, we know**
  - Transformers are universal function approximators.
  - Permutation invariance (robustness) is a function property.
  - Therefore, Transformers can approximate permutation-invariant functions.

Cause Analysis: Data Side

- **Suppose that**
  - we train the model for T epochs
  - the ICL number is N
  - we do data augmentation (e.g. shuffling opt) every epoch

  Can we solve the problem with data augmentation?

- **Then we need T >= N! to guarantee the worst-case robustness...**

Cause Analysis: algorithm Side

- **Empirical risk minimization**

$$\hat{\theta}_{\text{ERM}} := \arg\min_{\theta \in \Theta} \mathbb{E}_{(p,x,y)\sim\hat{P}}[\ell(\theta; p, x, y)]$$

- **Asymptotics properties of ERM:**

$$\hat{\theta} \xrightarrow{p} \theta^* \quad as\ n \to \infty$$

- **However, when data is limited…**

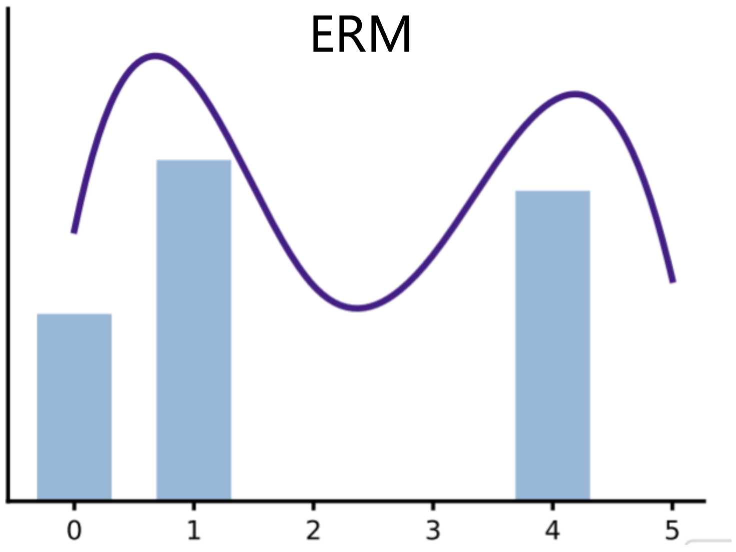


Figure 3. An illustrative example on 3 shot setting

How to design a method to mitigate the problems of ERM in the limited data setting?

PEARL: Permutation-Resilient Learning

The PEARL Objective

Optimize for the worst-case distribution Q_Π within an ambiguity set $\mathcal{Q}$:

$$\hat{\theta}_{\text{DRO}} \arg\min_{\theta\in\Theta}\left\{\sup_{Q_\Pi\in\mathcal{Q}} \mathbb{E}_{(p,x,y)\sim Q_\Pi}[\ell(\theta;p,x,y)]\right\}$$

Ambiguity Set $\mathcal{Q}$ (2/3)

The set of all possible permutations of the demonstrations in the empirical data:

$$\mathcal{Q} := \left\{\sum_{\Pi\in\mathbb{P}} q_\Pi\, Q_\Pi \,\middle|\, q \in \Delta_{|\mathbb{P}|-1}\right\}$$

where $Q_\Pi := \left\{(\Pi\cdot p,\, x,\, y) \,\middle|\, (p,x,y)\sim\hat{P}\right\}$.

- Π is a permutation matrix.
- $\mathbb{P}$ is the set of all $n!$ permutation matrices.
- q is a probability distribution over these permutations.

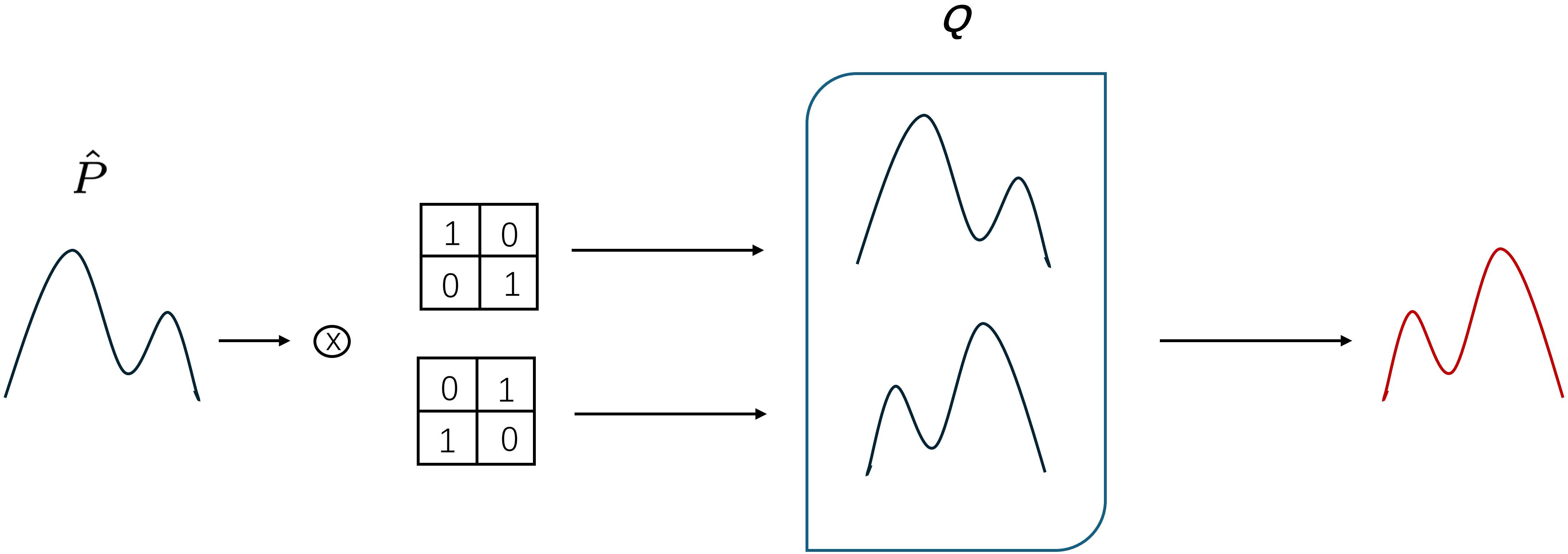
Intuition:

- ERM: optimize the avg loss
- PEARL: optimize the worst-case loss

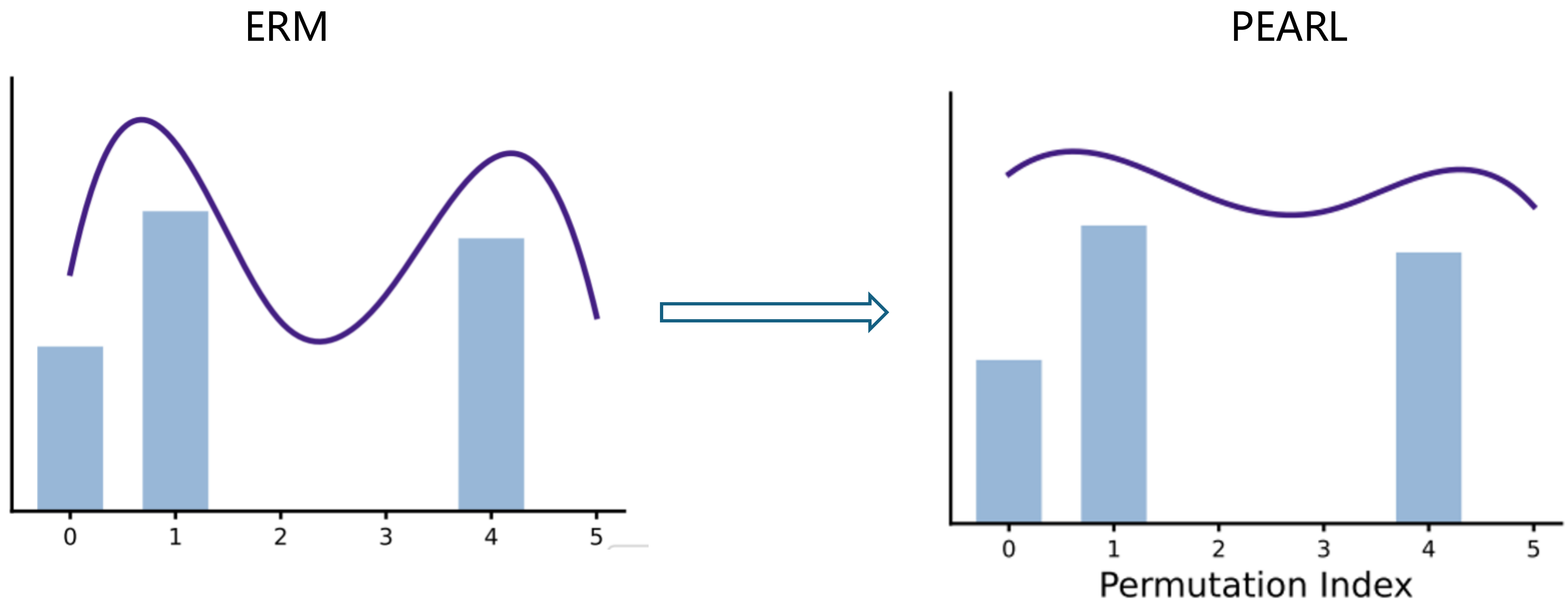
Q constaints all possible permutations of the empirical distribution.

Explanation of PEARL

- What happens in the inner problem?
  - Consider a 2 shot scenario



Comparsion of PEARL and ERM



How to effectively solve the problem?

- Notice that the outter problem can be solved by SGD, if the inner problem is solved. 🙂
- Solveing the inner problem (a selection problem) by brute force search will need O(n!). 🙁

Solving the Inner Problem via P-Net

- We design a permutation-proposal network (P-Net) to solve the inner problem.

$$(\mathcal{P} \times \mathcal{X} \times \mathcal{Y}) \rightarrow \Pi$$

a) ***Parameter componet:*** model the cross-relationship between n demons.

$$(\mathcal{P} \times \mathcal{X} \times \mathcal{Y}) \rightarrow \mathbf{R} \in \mathbb{R}^{n \times n}$$

b) ***Non-parameter component:*** get a permutation from the relationship representation.

$$\mathbf{R} \rightarrow \Pi \in \mathbb{R}^{n \times n}$$

Parameter Componet

- **Parameter componet:** model the cross-relationship between demons.

a) A feature extractor (an encoder model, e.g. BERT)

$$([\mathrm{CLS}], (x_1, y_1), \ldots, [\mathrm{CLS}], (x_n, y_n), [\mathrm{CLS}], (x, y)) \xrightarrow{\text{Encoder}} (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n, \mathbf{h}_{n+1}),$$

$$H = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n) \in \mathbb{R}^{n \times h}$$

b) A cross-relationship modeling layer (a MLP layer + non-linear activation)

$$\mathbf{R} = g\left(HWH^\top\right) \in \mathbb{R}^{n \times n}$$

Explanation of Parameter Component

What are the parameter componet doing? - A graph perspective

If we regard demonstrations as nodes in graph, then

- R is the adjacency matrix, Rij represent the relationship between demons i and j.
- The parameterized componet is doing an *edge predition task*

  e.g. larger Rij indicate we should swap the demons i and j (there is an edge)

So How to get a permutation from R? - Need futhuer operations…

Non-Parameter Componet

- **Non-parameter componet:** transform the adjacency matrix R into distribution, then sample a permutation from it.

a) *Sinkhorn algorithm:* transform R into a distribution over permutations. $\mathbf{R} \rightarrow \Delta(\Pi)$

$$S(R) = \lim_{l \to \infty} \left(\mathcal{T}_c \left(\mathcal{T}_r \left(\exp(R) \right) \right) \right),$$

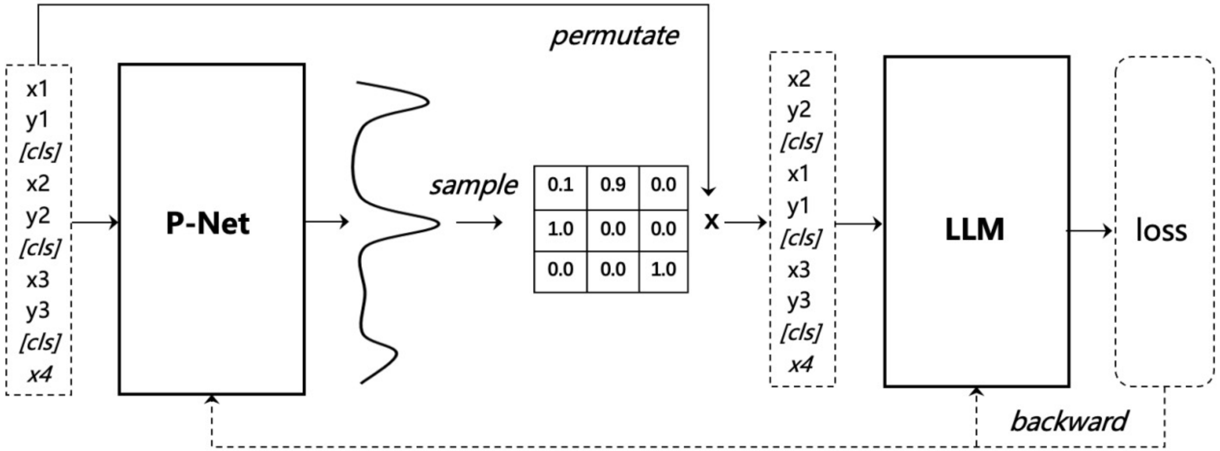
$$\mathcal{T}_r(R) = R \oslash \left(R \mathbf{1}_n \mathbf{1}_n^\top \right), \quad \mathcal{T}_c(R) = R \oslash \left(\mathbf{1}_n \mathbf{1}_n^\top R \right)$$

b) *Gumbel softmax:* sample a permutaiton from it. $\Delta(\Pi) \rightarrow \prod$

$$\Pi = \lim_{\tau \to 0} S\left((R + G)/\tau \right),$$

$$G_{ij} = -\log\left(-\log G'_{ij} \right), \quad G'_{ij} \sim U(0,1),$$

Adversarial Optimization



Algorithm 1: Adversarial Optimization Algorithm for PEARL

Input: θ, ϕ (LLM, P-Net); η_θ, η_ϕ (learning rates); m (inner steps); β (entropy coefficient)

repeat

  for $t = 1$ **to** m **do**

    $(p, x, y) \sim \hat{P}$; `// Sample training examples`

    $\Pi \sim \text{P-Net}(\phi, p, x, y)$; `// Generate permutations`

    $L_{\text{lm}}(\phi, \theta) \leftarrow \ell(\theta; \Pi \cdot p, x, y)$; `// Compute LLM loss`

    $L_{\text{ent}}(\phi) \leftarrow \mathcal{H}(\Pi)$; `// Compute entropy regularization`

    $\phi \leftarrow \phi + \eta_\phi \nabla_\phi (L_{\text{lm}} - \beta L_{\text{ent}})$; `// Update P-Net`

  end

  $\theta \leftarrow \theta - \eta_\theta \nabla_\theta L_{\text{lm}}(\phi, \theta)$; `// Update LLM`

until *convergence*;

Experiment

We Validate PEARL on Two Scenarios:

Scenario 1: ICL with Linear Functions

- Task: Pretrain a Transformer (GPT-2 base) from scratch to in-context learn $f(x) = w^\top x$.
- Metric: Normalized MSE.
- P-Net: BERT-base, trained from scratch.

Toy setting

Scenario 2: Instruction Tuning of LLMs

- Task: Fine-tune existing LLMs on Super-Natural Instructions (SNI).
- LLMs: Llama3-8B, Llama2-7B/13B, Mistral-7B, Gemma-7B.
- P-Net: FLAN-large encoder.

Realistic setting

Results

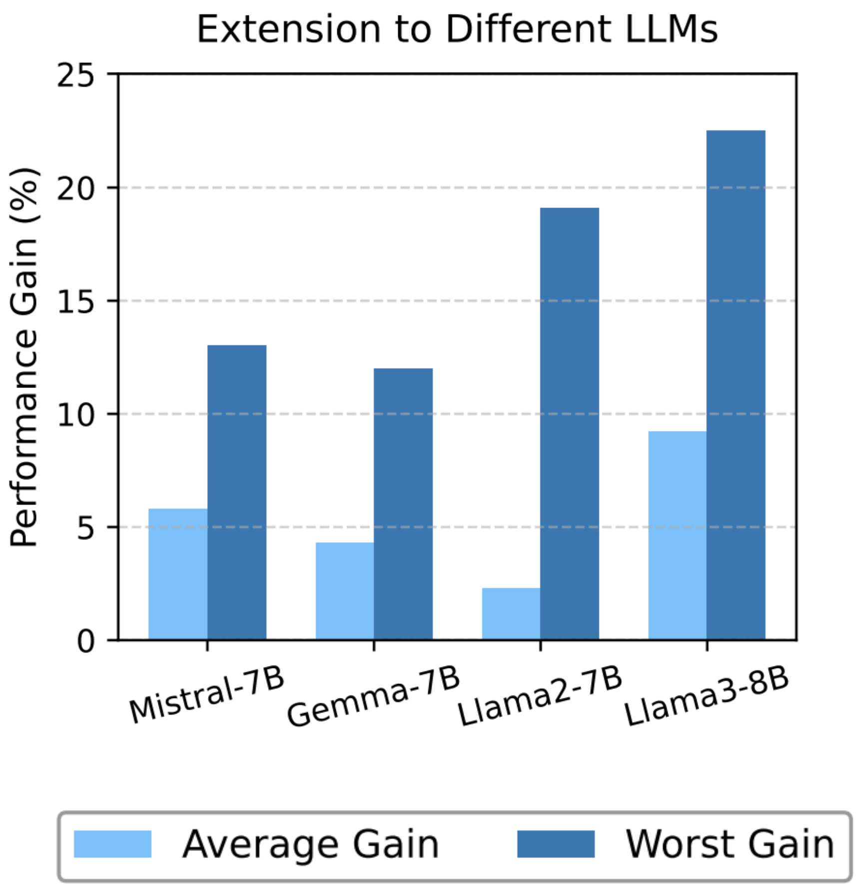
- Performance on different LLMs

P-Net

- Flan-large encoder

LLMs

- SOTA base model 7~8B



Results

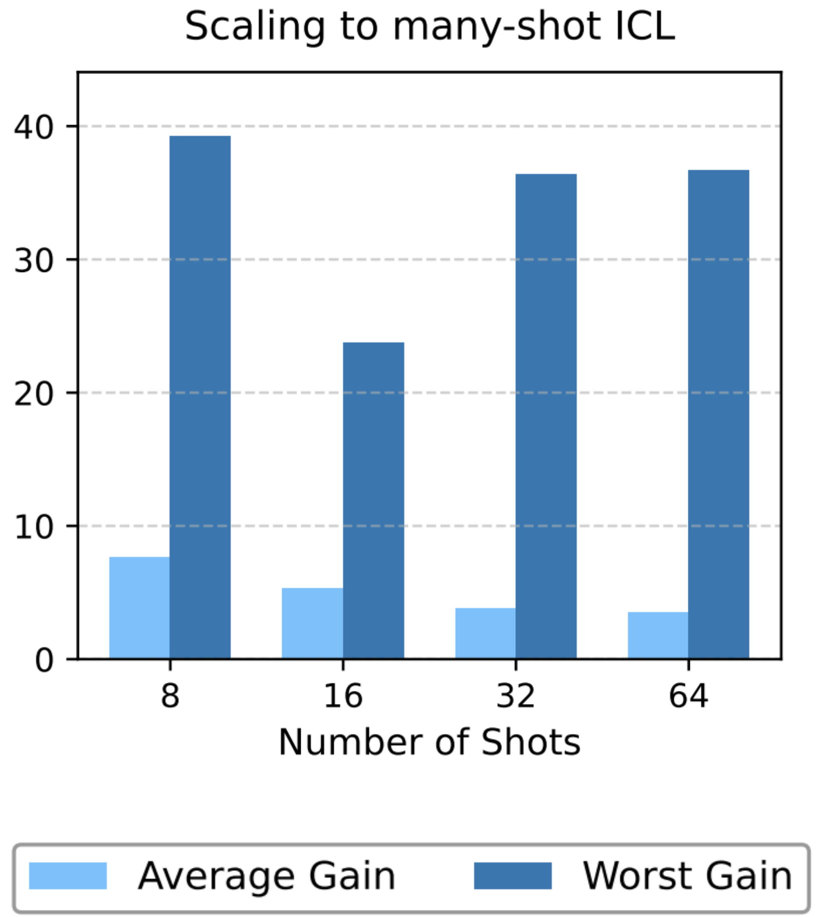
- Generalize to many-shot, long-context setting.

Train

- 5 shot
- 512 seq length

Test

- 8 ~ 64 shot
- 8k seq length



Results

- Improves shot efficiency: #shots that ERM requires to match the avg performance of PEAEL.

PEARL needs 2 to 4 times fewer shots.

Table 4: Shot Efficiency: Average Performance with and without PEARL.

# Shots	2	4	8	16	32	64
ERM	57.3	59.7	61.8	66.9	67.4	68.1
PEARL	62.9	63.1	66.5	70.5	70.0	70.4

An Unexpected Benefit: Improved Best-Case Performance

- Improves shot efficiency: #shots that ERM requires to match the avg performance of PEAEL

Table 12: Best performance comparison between ERM and PEARL

#Shot	Method	Average	Gain	CSQA	CurDial	CoLA	TMW
2	ERM	64.1	-	68.8	64.4	64.1	59.2
	PEARL	68.8	7.2%	73.4	69.2	70.3	62.1
3	ERM	72.8	-	70.3	85.0	65.6	70.3
	PEARL	77.0	5.7%	73.4	87.9	79.7	66.9
4	ERM	82.9	-	81.3	92.4	78.1	79.7
	PEARL	84.3	1.7%	82.8	93.6	81.2	79.5
5	ERM	86.8	-	84.4	95.3	81.3	86.2
	PEARL	89.3	2.9%	87.5	96.5	85.9	87.3

PEARL: Summary Future Outlook

- **Summary**

a) Mitigate the shortcomings of ERM in the limited data setting
b) Designed a neural solution approach (P-Net, Learn to permute) for the inner problem
c) Provides a general framework for handling set-structured inputs with order-independent elements

- **Future Directions**

  - Multiple documents: "Lost in the middle" problem
  - Multiple images or videos…

Thank You!

Q & A