PEARL: TOWARDS PERMUTATION-RESILIENT LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The in-context learning (ICL) ability of large language models (LLMs) enables them to undertake challenging tasks using provided demonstrations. However, it is prone to instability: different orderings of demonstrations can significantly influence predictions, revealing LLMs' limitations in processing combinatorial inputs. This paper shows that this vulnerability can be exploited to design a natural attack that is imperceptible to the model provider and can achieve nearly 80% success rates on the SOTA open-source model, LLaMA, by simply permuting the demonstrations. In light of this, how to overcome the ordering sensitivity problem is an important issue for improving the performance of LLMs. However, current mitigation methods focus on post-processing and fail to enhance models' inherent robustness to the vast space of possible input permutations. To overcome this issue, we propose a novel **Pe**rmut**a**tion-**r**esilient **l**earning framework (**PEARL**) based on distributionally robust optimization (DRO), which optimizes model performance against the worst case among all possible permutations. Specifically, PEARL consists of a hard permutation mining network (P-Net) and the LLM. The P-Net identifies the most challenging permutations by formulating the task as an optimal transport problem, which is solved using an entropy-constrained Sinkhorn algorithm. Through minimax optimization, the P-Net progressively generates harder samples to enhance the LLM's worst-case performance. Experiments with synthetic data and instruction tuning tasks demonstrate that the PEARL framework effectively mitigates permutation attacks and improves overall performance.

1 INTRODUCTION

A hallmark of human intelligence is the ability to learn and execute new tasks by reasoning from a few examples. Mirroring this, in-context learning (ICL) (Brown et al., 2020), as a crucial supplement to zero-shot prompting, has shown promising results across a spectrum of complex tasks (Cobbe et al., 2021; Chowdhery et al., 2023; OpenAI et al., 2023). Despite these advancements, the ICL capabilities of large language models (LLMs) remain fragile. LLMs exhibit sensitivity to permutations of provided demonstrations (Lu et al., 2022; Zhao et al., 2021; Reynolds & McDonell, 2021). This fragility underscores a significant gap in achieving human-like adaptability.

Most existing studies on ICL primarily aim to enhance the normal-case performance on few-shot learning (Min et al., 2022; Wei et al., 2023), with limited attention to improving permutation robustness. Current strategies addressing this issue in few-shot learning generally fall into two categories: 1) Output Calibration (Zhao et al., 2021), which proves effective for classification tasks but is less applicable to generation tasks, and 2) Order Optimization (Lu et al., 2022), which focuses on finding the optimal sequence of few-shot demonstrations during inference but suffers from exponential computational complexity. Consequently, there remains a significant need for methods that can fundamentally enhance LLMs' inherent ability to manage the vast combinatorial space of possible input permutations.

In this work, we first conduct extensive experiments on LLaMA-3 to revisit the vulnerability of latest LLMs to permutations of ICL (§3). Our empirical analysis reveals that even state-of-the-art open-source LLMs, such as LLaMA-3-8B, are still highly susceptible to a simple permutation-based attack that merely alters the order of ICL demonstrations. Remarkably, these attacks, which do not modify the semantic content of the examples or append any malicious suffixes, can achieve success rates exceeding 80%. Consequently, these attacks are less noticeable to model providers but highly effective against LLMs, highlighting a critical vulnerability of LLms.

To counteract the vulnerability to input permutations, we introduce a novel **Pe**rmut**a**tion-**r**esilient **l**earning (**PEARL**) framework, which is based on distributionally robust optimization (DRO) (Ben-Tal et al., 2011). Unlike standard empirical risk minimization training, adopted by most supervised fine-tuning (SFT) methods, which views each training instance merely in terms of its one or several permutations observed during training, DRO conceptualizes each instance as part of a broader distribution that includes all conceivable permutations. This comprehensive set of all possible permutations is termed the ambiguity set. By explicitly identifying and optimizing the worst-case within this ambiguity set, our strategy substantially enhances the resilience of LLMs against all different permutations. This paradigm shift—from considering training instances as single data points to viewing them within a distribution of potential permutations— equips the model to better prepare for and generalize to combinatorial input scenarios.

Specifically, PEARL operationalizes DRO as a two-player game, consisting of a hard permutation mining network (P-Net) as the adversary and the LLM as the target model. For each training instance, P-Net identifies a hard permutation of given demonstrations, aiming to maximize the LLM's loss. Conversely, the LLM strives to minimize its loss under the P-Net's perturbations, thereby performing well on these challenging examples. P-Net frames the identification of the most adversarial ICL permutation as an optimal transport (OT) (Monge, 1781) problem between the uniform distribution over permutations and the distribution of currently challenging permutations. We solve the OT problem using the Sinkhorn algorithm (Sinkhorn, 1966) with an element-wise entropy constraint designed to prevent trivial solutions. Through adversarial training (AT), both networks improve iteratively. Ideally, at convergence, the P-Net represents a uniform distribution across all permutations, as the LLM handles all possible permutations equally well.

We validate our method in two widely used scenarios: (1) pre-training a transformer to in-context learn linear functions, and (2) instruction finetuning of LLMs on real-word tasks. Comprehensive evaluations demonstrate that compared to ERM-based training, our method consistently and substantially improves both the average and worst-case performance of LLMs across all possible permutations and effectively defends against permutation-based attacks. Notably, in practical instruction tuning scenarios, our method achieves superior results with only hundreds of LoRA parameter updates, highlighting its exceptional effectiveness and efficiency.

2 RELATED WORK

Order Sensitivity in In-context Learning Despite the huge success of ICL, its robustness to demonstration permutations remains an unresolved challenge (Zhao et al., 2021). Most ***training-stage methods*** focus on improving general performance in ICL (Min et al., 2022; Wei et al., 2023) while neglecting the lack of robustness to the permutations of demonstrations. Recent studies suggest that this phenomenon stems from the autoregressive nature of transformer language models (Chen et al., 2023; Xiang et al., 2024). InfoAC (Xiang et al., 2024) introduces contrastive learning during fine-tuning to break the autoregressive constraint and enable bidirectional token visibility; however, their approach achieves limited success and is restricted to classification tasks. Preliminary work of (Chen et al., 2023) shows the DeepSet architecture exhibits better permutation invariance than transformer; however, this MLP-based new architecture is too small to solve complex language modeling tasks. ***Inference-stage methods*** can be categorized into four types: (1) demonstration selection (Chang & Jia, 2023; Peng et al., 2024), which primarily enhances normal-case performance without guaranteeing worst-case performance under permutations; (2) output calibration (Zhao et al., 2021; Li et al., 2023; Guo et al., 2024a), which proves effective for classification tasks but is less applicable to generation tasks due to sequence calibration challenges; (3) order optimization (Lu et al., 2022), which aims to find the best ordering during inference but suffers from exponential computational complexity; and (4) prediction ensembling: a recent work (Zhang et al., 2024) proposes to transform an n-shot ICL into n one-shot predictions and ensembles the results—this is effective for classification but leads to decreased performance on generation tasks. *In summary, In summary, inference-stage methods aims to circumvent order sensitivity by pre/post-processing without fundamentally enhancing the robustness of LLMs to different orders. Moreover, most methods are designed for classification tasks and show reduced effectiveness on generation tasks.* To the best of our knowledge, our work is the

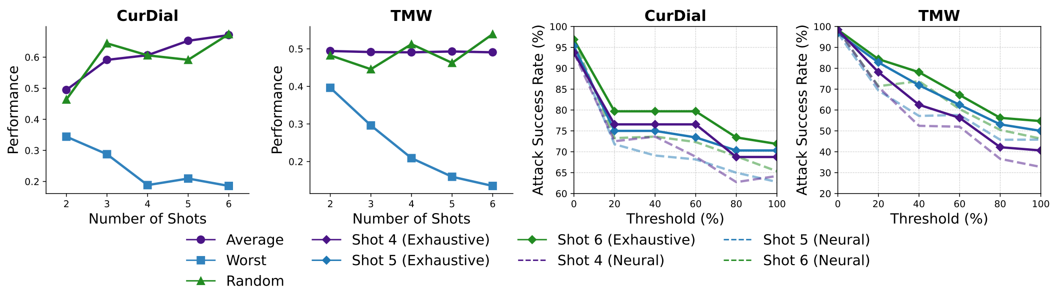

Figure 1: Performance and attack success rates of Llama-3 on CurDial and TMW datasets. Left panels: Random, average and worst-case performance as a function of shot number. Right panels: Attack success rates for exhaustive and neural search attack methods at different thresholds.

first to solve this problem from an adversarial perspective. We propose a novel distributionally robust optimization (DRO)-based learning algorithm to enhance the inherent robustness of LLMs against order perturbations and solve it using the Sinkhorn operator. Our approach complements existing inference-stage methods and generalizes across diverse task categories.

Distributionally Robust Optimization. In distributionally robust optimization (DRO), ambiguity sets are often defined as divergence balls centred on the empirical distribution of data pairs (x, y), which act as regularizers for small radii (Ben-Tal et al., 2013; Lam & Zhou, 2015; Duchi et al., 2016; Miyato et al., 2018). However, larger radii can result in excessively conservative sets. Prior applications of DRO have addressed distributional shifts, including label shift (Hu et al., 2018) and data source shift (Oren et al., 2019) and group shift (Sagawa et al., 2020). In contrast, this study is the first to apply DRO to in-context learning robustness, defining the ambiguity set through all possible permutations of the empirical distribution that requires ICL performance guarantees.

Optimal Transport. Optimal transport (OT), a foundational mathematical discipline established by (Monge, 1781; Kantorovich, 1942), provides a metric for measuring distances between distributions, commonly known as the Wasserstein distance or Earth Mover Distance. It has been applied as a tool for manipulating probability distributions. In our study, the hard Permutation mining Network (P-Net) is designed to act as a conduit for transportation between two discrete measures, leveraging entropy-constrained OT (Cuturi, 2013), also referred to as the Sinkhorn distance, to enable the derivation of a differentiable loss (Genevay et al., 2018). Our work extends the concept of learning permutation structures through neural networks, as explored in (Mena et al., 2018) for learning to sort numbers or solve jigsaw puzzles. However, we apply OT in the context of LLMs, and design a neural network (P-Net) equipped with Sinkhorn operator to generate challenging permutations for LLMs to perform adversarial training.

3 Revisiting Permutation Vulnerability in LLMs

This section examines the severity of performance fluctuations in LLMs in response to different permutations of given demonstrations. Additionally, from an adversarial perspective, we explore whether this vulnerability can be exploited to devise an imperceptible and effective attack on LLMs.

Experimental Setups To conduct evaluations, we select two tasks from Super-NaturalInstructions (Wang et al., 2022), including Curiosity-based Dialog (CurDial) and TellMeWhy QA (TMW). We test 100 samples for each task, with each sample structured as a quadruple consisting of (instruction, demonstrations, input, output). The number of demonstrations (shots) ranges from two to six. Following (Wang et al., 2022), the performance is measured using the ROUGE-L (Lin, 2004). We adopt LLaMA-3-8B for evaluation due to its widespread use. We analyze the permutation vulnerability of LLaMA-3-8B on two settings as follows:

1) Permutation Vulnerability on Different Number of Demonstrations We first examine the average and worst-case performance of the model across different permutations of input demonstrations and the effect of scaling the number of demonstrations. As shown in the left of Figure 1, there is a notable observation: ***adding demonstrations is a double-edged sword.*** Increasing the number of demonstrations (*shots*) generally enhances the model's average performance due to richer contextual information. However, it can simultaneously worsen the worst-case performance. This suggests that while more demonstrations provide beneficial context, the exponentially increasing number of possible permutations (n!) introduces a higher likelihood of a possible input configuration on which the model performs poorly.

2) Input Permutation as Attack We then consider a two-party adversarial attack scenario between a malicious user (attacker) and a model provider (defender). The malicious user aims to induce compromised response by solely permuting the in-context demonstrations, which is less noticeable for the model provider. We measure the effectiveness of such attack by reporting the attack success rate (ASR). Given a task $D = \{(p_i, x_i, y_i)\}$, we define a sample (p_i, x_i, y_i) successfully attacked if its relative performance degradation induced by a attacher exceeds a threshold $\delta \in [0\%, 100\%]$. Here, p_i represents an ICL prompt containing n demonstrations. We denote the set of all possible permutations of the p_i demonstrations as $\mathbb{P} = \{\Pi_0, \ldots, \Pi_{n!-1}\}$, where $|\mathbb{P}| = n!$. Let g be a performance metric function (e.g., ROUGE-L). The ASR for a dataset D is defined as:

$$\text{ASR}(D, \delta) = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathbb{I}\left(\frac{|\mu_i - \omega_i|}{\mu_i} \geq \delta\right) \tag{1}$$

where $\mathbb{I}$ denotes the indicator function, $|D|$ is the size of the dataset, and δ is the threshold. The average performance of the i-th sample, μ_i, is defined by:

$$\mu_i = \mathbb{E}_{\Pi \sim \mathbb{P}}[g(\Pi \cdot p_i, x_i; y_i)] = \frac{1}{n!} \sum_{j=1}^{n!} g(\Pi_j \cdot p_i, x_i; y_i) \tag{2}$$

and ω_i is the compromised performance induced by the attack strategy adopted by the malicious user. Here, we analyze two attack method:

- **Exhaustive Search Attack**: To calculate the upper bound of the effect the permutation-based attack can achieve, we assume that the malicious user has unlimited attempts and conducts an exhaustive search. For each sample (p_i, x_i, y_i), this process involved testing all possible permutations of demonstrations in Q_i and identifying the permutation that yields the poorest performance. In this case, the attacked performance is calculated as follows:
$$\omega_i = \min_{\Pi \in \mathbb{P}} g(\Pi \cdot p_i, x_i; y_i) \tag{3}$$
- **Neural Search Attack**: When the number of attempts is limited, we employ a meta-learning method to optimize a hard permutation mining network (P-Net), aiming to approximate the upper bound established by the exhaustive search. As shown in Figure 3 (details are in the method section), during training, this network takes the standard sample (p_i, x_i, y_i) as input and outputs a permutation matrix Π_i. The permuted samples $(\Pi_i \cdot p_i, x_i, y_i)$ are then fed into the LM to maximize its loss function. During testing, the network generates the most challenging permutation Π_i for each sample (p_i, x_i, y_i). Then the attacked performance is calculated as follows:
$$\omega_i = g(\Pi_i \cdot p_i, x_i; y_i), \qquad \Pi_i = \text{P-Net}(p_i, x_i, y_i) \tag{4}$$

As shown in the right of Figure 1, the results indicate that ***permutation attacks are effective and approachable.*** Leveraging this characteristic, the exhaustive search attack successfully attacks over 50% and 80% of the samples with $\delta = 50\%$ on two datasets respectively, and the neural attack achieved a successful rate close to this upper bound across different shots. These results demonstrate that this vulnerability poses a real concern, even for advanced LLMs like LLaMA-3.

Remark These deficiencies may directly stem from the fundamental limitations of standard Empirical Risk Minimization (ERM) training, which focuses on optimizing average performance while neglecting worst-case performance. We discuss this issue in depth in the next section and propose a method to address the model's improper behaviour on unseen but practically valid input spaces.

4 PERMUTATION-RESILIENT LEARNING (PEARL)

4.1 INSTRUCTION TUNING VIA DRO

Our objective is to train a LLM to perform well across all possible permutations of given demonstrations when prompted with few-shot instructions.

In supervised fine-tuning for few-shot learning, the LLM is trained to predict an output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$ and a few-shot instruction $p \in \mathcal{P}$, where p typically consists of a sequence of demonstrations, each being an input-output pair. Let Θ denote the parameter space of the language model, and let $\ell : \Theta \times (\mathcal{P} \times \mathcal{X} \times \mathcal{Y}) \to \mathbb{R}_+$ be a nonnegative loss function measuring the discrepancy between the model's prediction and the true output. The standard approach is to find parameters $\theta \in \Theta$ that minimize the empirical loss over the training data via empirical risk minimization (ERM):

$$\hat{\theta}_{\text{ERM}} := \arg\min_{\theta \in \Theta} \mathbb{E}_{(p,x,y)\sim \hat{P}}[\ell(\theta; (p, x, y))] \tag{5}$$

where $\hat{P}$ denotes the empirical distribution derived from the training dataset.

Under appropriate assumptions, learning theory (Vapnik, 1999; Shalev-Shwartz & Ben-David, 2014) guarantees that models trained via ERM perform well on the test distribution given sufficient training data. However, in practice, models trained using ERM often fail to generalize well to different permutations of the same set of demonstrations. This occurs because the training set covers only a subset of all possible permutations of the demonstrations, and during testing, the model may encounter permutations not seen during training, leading to a significant degradation in performance.

To systematically address the permutation sensitivity issue, we propose fine-tuning under the framework of **distributionally robust optimization (DRO)**, which optimizes the risk under the worst-case distribution within a specified ambiguity set. Specifically, we aim to solve:

$$\hat{\theta}_{\text{DRO}} = \arg\min_{\theta \in \Theta} \Big\{ \sup_{Q_\Pi \in \mathcal{Q}} \mathbb{E}_{(p,x,y)\sim Q_\Pi}[\ell(\theta; (p, x, y))] \Big\} \tag{6}$$

The ambiguity set $\mathcal{Q}$ is constructed to capture all distributions obtained by permuting the prompts in the empirical distribution $\hat{P}$. Specifically, for each possible permutation $\Pi \in \mathbb{P}$, we define the permuted distribution Q_Π by applying Π to the prompt p of each data point in $\hat{P}$:

$$Q_\Pi := \Big\{ (\Pi \cdot p,\ x,\ y) \,\Big|\, (p, x, y) \sim \hat{P} \Big\}, \quad \Pi \in \mathbb{P}, \tag{7}$$

where Π is a permutation matrix acting on the sequence of demonstrations in p, and $\mathbb{P}$ denotes the set of all possible permutation matrices. The ambiguity set $\mathcal{Q}$ is then defined as the convex hull of these permuted distributions:

$$\mathcal{Q} := \Bigg\{ \sum_{\Pi \in \mathbb{P}} q_\Pi \, Q_\Pi \,\Bigg|\, q \in \Delta_{|\mathbb{P}|-1} \Bigg\}, \tag{8}$$

where q is a probability vector belonging to the $|\mathbb{P}| - 1$-dimensional simplex $\Delta_{|\mathbb{P}|-1}$.

By considering all possible permutations of the prompts in the empirical distribution, $\mathcal{Q}$ encompasses all distributions that could arise due to prompt permutations. This formulation allows DRO to identify the worst-case distribution within $\mathcal{Q}$ (the $\sup$ step in Eq. 6) and optimize the model's performance against it (the $\arg\min$ step), thereby enhancing robustness to permutations in the input data.

To illustrate the advantages of DRO over ERM in handling different permutations, consider the example in Figure 2. For a 3-shot training example (p, x, y) with prompt p containing three demonstrations, there are six possible permutations denoted as $(p^0, x, y), \ldots, (p^5, x, y)$, indexed from 0 to 5. $\hat{P}$ denotes the empirical distribution of permutations in training data, represented by blue bars. The bars show that permutation indices 0, 1, and 4 appear in training data with frequencies, while permutations 2, 3, and 5 do not appear. P_θ represents the distribution learned by the LLM, represented by purple curves. In panel (a), the ERM-trained model assigns higher probabilities to frequently occurring permutations $(0, 1, 4)$ and lower probabilities to less frequent ones $(2, 3, 5)$, leading to poor performance on unseen permutations during testing. In contrast, panel

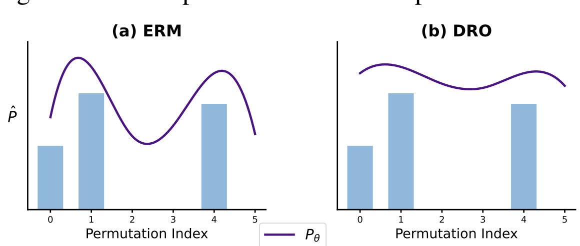

Figure 2: Comparison of models trained under ERM and DRO paradigms. The blue bars represent the empirical distribution $\hat{P}$ of training data, showing different frequencies of six permutations in the training set. The purple curves denote the learned distribution P_θ by (a) ERM and (b) DRO models, illustrating their different behaviors on less appeared but valid permutations.

(b) shows that the DRO-trained model distributes probabilities more uniformly across all possible permutations, as it explicitly considers them all (Equation (6)) during learning. This demonstrates how DRO mitigates ERM's limitations by encouraging models to assign reasonable probabilities to all valid permutations, regardless of their frequency in training data.

4.2 P-Net: Learning To Permute via Optimal Transport

To enable our DRO framework to function effectively, we need to efficiently find the worst-case scenario within the ambiguity set (solve the max step in Equation (6). Directly addressing this problem through exhaustive search is computationally infeasible due to the exponential search space.

To overcome this challenge, we pose the following question: *How can we transform a distribution of the permutations of the given demonstrations into a target distribution that satisfies desired properties, such as worst-case permutations for LLMs?* We address this problem by modeling it as an optimal transport (OT) problem and solving it using the Sinkhorn algorithm.

We introduce a neural network called the **Hard Permutation Mining Network (P-Net)**, a dedicated model designed to generate permutations that are challenging for the LLM. As illustrated in Figure 3, P-Net functions by inputting all demonstration pairs contained within a prompt (input permutation) and outputs a permutation matrix (target permutation), which is then used to reorder the demonstrations into a harder version. The P-Net comprises three components: feature extraction, cross-demonstration interaction, and the Sinkhorn operator.

Specifically, the feature extractor of the P-Net can be a small pre-trained transformer that takes a ICL prompt composed of n demonstration pairs $p = \{(x_i, y_i)\}_{i=1}^{n}$ and a predicting sample (x, y), and produces their representations as fowllows:

$$([\text{CLS}], (x_1, y_1), \ldots, [\text{CLS}], (x_n, y_n), [\text{CLS}], (x, y)) \xrightarrow{\text{Transformer}} (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n, \mathbf{h}_{n+1}), \tag{9}$$

where $\mathbf{h}_i$ is the representation corresponding to the i-th [CLS] token, which is often used to segment and extract the representation of sequences (Devlin et al., 2019b; Lu et al., 2021).

To model the pairwise relationships among the demonstrations, we define $H = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$ in $\mathbb{R}^{n\times h}$, and apply a cross-demonstration interaction operation to obtain a relationship matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$, defined as

$$\mathbf{R} = g\left(HWH^\top\right), \tag{10}$$

where $H \in \mathbb{R}^{n\times h}$ contains the representations of N demonstrations, $W \in \mathbb{R}^{h\times h}$ is a weight matrix, and g denotes a nonlinear activation function.

The output matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ can be interpreted as an **adjacency matrix** in graph theory. Viewing the demonstrations as nodes in a graph, the relationship between nodes i and j is represented by the edge R_{ij}. Specifically, the element R_{ij} indicates the potential increase in difficulty for the LLM if demonstrations i and j are swapped; a larger value of R_{ij} suggests that swapping these two demonstrations may significantly increase the difficulty for the LLM.

Note that we ultimately aim for P-Net to output a permutation matrix; however, the elements of $\mathbf{R}$ can take any real values at this stage. To further enforce that the proposed matrix $\mathbf{R}$ approximates a doubly stochastic matrix—a square matrix in which all rows and all columns sum to one and all elements are non-negative—we employ the **Sinkhorn operator** S. The Sinkhorn operator is a well-established method in optimization theory that transforms a square matrix into a doubly stochastic matrix through iterative row and column normalization (Sinkhorn, 1966; Adams & Zemel, 2011; Mena et al., 2018). The iterative process for a matrix R is defined as:

$$S(R) = \lim_{l\to\infty} \left(\mathcal{T}_c\left(\mathcal{T}_r\left(\exp(R)\right)\right)\right), \tag{11}$$

$$\mathcal{T}_r(R) = R \oslash \left(R\mathbf{1}_n\mathbf{1}_n^\top\right), \quad \mathcal{T}_c(R) = R \oslash \left(\mathbf{1}_n\mathbf{1}_n^\top R\right), \tag{12}$$

where $\mathcal{T}_r(R)$ and $\mathcal{T}_c(R)$ represent the row and column normalization operators, respectively; $\oslash$ indicates element-wise division; and $\mathbf{1}_n$ is a column vector of ones. As established by (Sinkhorn, 1966), the Sinkhorn operator $S(R)$ strictly converges to a doubly stochastic matrix as the number of iterations l approaches infinity.

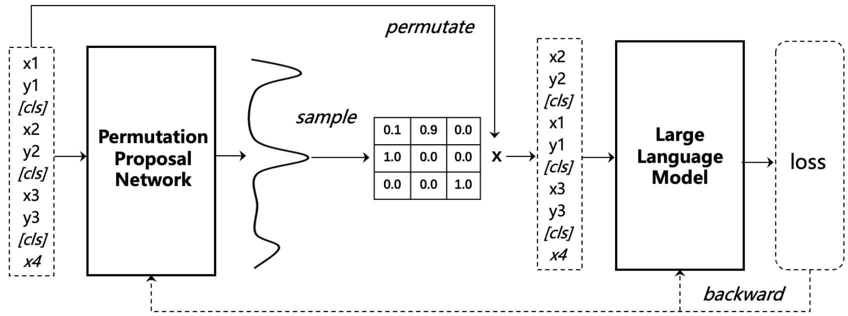

Figure 3: An overview of the learning framework. The P-Net is a small model incorporating optimal transport (OT) algorithm, trained jointly with the LLM under the adversarial optimization paradigm. *Note that the permutation matrix operates on the input sequence's embeddings (simplified here as text sequences for clarity).* After training, only the LLM is retained while the P-Net is discarded.

To sample a permutation matrx from the double stochastic matrix (Figure 3) and build a differentiable process, the gumbel sampling (Jang et al., 2017) is applied to the sinkhorn operator:

$$\Pi = S\left(\frac{R+U}{\tau}\right), \quad U_{ij} = -\log\left(-\log\left(U'_{ij}\right)\right), \quad U'_{ij} \sim \text{Uniform}(0,1), \tag{13}$$

where $U \in \mathbb{R}^{n\times n}$ is a matrix of Gumbel noise and τ is temperature. As τ approaches zero, $S\left((R+U)/\tau\right)$ approximates a permutation matrix $\Pi \in \mathbb{P}^{n\times n}$. The hyperparaters of the Sinkhorn operator are studied in Appendix D.

By modelling permutation generation as an OT problem and designing the P-Net to implement it, we enable the transformation of the input permutation distribution into a target permutation distribution. Next, we introduce how P-Net is co-trained with LLM so that the obtained target permutation distribution satisfies the desired property—being the most challenging for the current LLMs.

4.3 Adversarial Optimization

As depicted in Figure 3, our framework employs an adversarial approach to co-optimize the LLMs and the P-Net. Specifically, for each sample, the P-Net generates a challenging permutation designed to maximize the LLM's loss. In turn, the LLM seeks to minimize its loss despite the challenging permutations introduced by the P-Net. Let θ denote the parameters of the LLM, and ϕ those of the P-Net. We formalize the optimization process as follows.

We first optimize the P-Net, corresponding to the inner maximization step in Equation (6). For a given example (p, x, y), we sample a permutation $\Pi \sim \text{P-Net}(\phi; (p, x, y))$ from P-Net. We then compute the LLM's loss on the permuted example $(\Pi \cdot p, x, y)$, denoted by $\ell(\theta; \phi; (\Pi \cdot p, x, y))$. The objective is to optimize the P-Net parameters ϕ to maximize this loss:

$$L(\phi;\theta)_{lm} = \mathbb{E}_{(p,x,y)\sim\hat{P},\Pi\sim\text{P-Net}(\phi;(p,x,y))}[\ell(\theta;\phi;(\Pi\cdot p,x,y))] \tag{14}$$

Note that the Sinkhorn operator is implicitly included in $\Pi \sim \text{P-Net}(\phi; (p, x, y))$.

To prevent the P-Net from exploiting trivial solutions, such as outputting uniform matrices that dilute the semantic content of the demonstrations, we introduce an element-wise entropy constraint term that encourages Π to be as distinct as possible:

$$L(\phi)_{ent} = \mathbb{E}_{(p,x,y)\sim\hat{P},\Pi\sim\text{P-Net}(\phi;(p,x,y))} \sum_{i,j} \Pi_{ij}(1-\Pi_{ij}). \tag{15}$$

This leads to the following combined optimization for the P-Net:

$$\hat{\phi}^{\star} = \arg\max_{\phi\in\Phi} \left(L(\phi;\theta)_{lm} - \beta L(\phi)_{ent}\right), \tag{16}$$

where β represents the penalty coefficient for the entropy constraint, further studied in Appendix D. **Note when optimizing Equation (16), θ remains constant.**

We then optimize LLM, corresponding to the inner minimization step in Equation (6). For an example (p, x, y), *we get a challenging permutation from the previously optmized P-Net* ($\hat{\phi}^\star$), $\Pi \sim$ *P-Net*$(\hat{\phi}^\star; (p, x, y))$. We compute the LLM's loss on this permuted example $(\Pi \cdot p, x, y)$, denoted by $\ell(\theta; \hat{\phi}^\star; (\Pi \cdot p, x, y))$. The objective is to optimize the LLM parameters θ to minimize this loss:

$$\hat{\theta}^\star = \arg\max_{\theta \in \Theta} L(\hat{\phi}^\star; \theta)_{lm}, \tag{17}$$

Note when optimizing LLM, we incorporate the previously optimized parameter $\hat{\phi}^\star$ from the P-Net and keep it constant.

From Equation (16) to (17), we complete a loop of iteration. In the next iteration, we substitute $\hat{\theta}^\star$ into Equation (16) for a new round of optimization until convergence. The comprehensive training algorithm is outlined in Appendix A.

5 In-Context Learning with Linear Functions

5.1 Datasets and Evaluation Metrics

We investigate in-context learning on linear functions $f(x) = w^\top x$, where $w \in \mathbb{R}^d$, following (Garg et al., 2022; Guo et al., 2024b). For each w, we construct prompts $p^i = (x_1, f(x_1), \ldots, x_i, f(x_i), x_{i+1})$ containing i input-output demonstration pairs and a query input x_{i+1}. A language model LM_θ is trained to minimize:

$$\min_\theta \mathbb{E}_p \left[\frac{1}{k+1} \sum_{i=0}^{k} \ell(LM_\theta(p^i), f(x_{i+1})) \right], \tag{18}$$

where $\ell(\cdot)$ is the MSE loss and k is the maximum number of demonstrations. We evaluate using normalized squared error $((LM_\theta(p) - w^\top x_{\text{query}})^2/d)$. Detailed settings are in Appendix B.1.

5.2 Implementation Details and Baselines

Architecture and Training. We implement L_θ using a GPT-2 base model (Radford et al., 2019) and train it from scratch on a generated dataset using the AdamW (Loshchilov & Hutter, 2019). Key training parameters include a batch size of 128 and 500k training steps. In the PEARL framework, the P-Net is initialized as a BERT-base (Devlin et al., 2019a) and also trained from scratch. Implementation details are in Appendix B.2.

Baselines. Consistent with (Garg et al., 2022), we adopt an empirical risk minimization method with curriculum learning (Bengio et al., 2009; Wu et al., 2020) **(ERM+CL)** to train the model. The training process gradually increase the number of demonstrations presented to the model, allowing for progressive learning of more complex patterns and making the training more stable.

5.3 Evaluation Results

We evaluate the effect of permutations on the worst-case and average performance of different methods, as well as each method's defence capability against permutation attacks.

As shown in Table 1, the performance gap between average and worst-case performance across permutations for the baseline methods was significant, indicating substantial vulnerability to permutations. Specifically, the worst-case performance of the baseline methods decreased dramatically compared to their average performance, with the relative performance drop increasing from 74.6% at 3 shots to 84.1% at 4 shots, effectively losing most of the performance gains achieved by increasing the number of shots. In contrast, our method, PEARL, not only improved the average performance but also significantly enhanced the worst-case generalization performance compared to the baselines. While the average performance gains tend to plateau as the number of shots increases, the worst-case performance gains continue to rise, increasing from 65.5% at 3 shots to 73.6% at 5 shots.

Figure 4 depicts the proportion of successfully attacked samples in terms of (1) different attack success thresholds and (2) number of demonstrations (shots). The former considers more pessimistic scenarios (attacked samples drop a large margin), while the latter examines larger input spaces. We

Table 1: Normalized MSE across permutations.

Shot	Method	Avg.	Worst.
3	ERM+CL	1.45	2.67
	PEARL	**0.86** (+40.7)	**0.92** (+65.5)
4	ERM+CL	1.20	3.34
	PEARL	**0.79** (+34.1)	**1.11** (+66.8)
5	ERM+CL	1.28	5.03
	PEARL	**0.87** (+32.0)	**1.33** (+73.6)

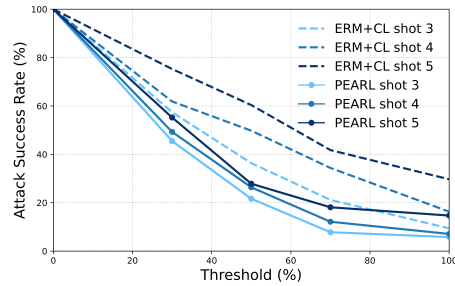

Figure 4: Comparison of attack success rates.

observed that PEARL's advantage increased as the threshold grew. At $\delta > 50\%$, the defence success rate for PEARL across all shots was approximately double that of the baseline methods. This indicates that PEARL can effectively prevent pessimistic scenarios (samples attacked with a large threshold). Moreover, PEARL's performance improved with an increasing number of shots, suggesting better scalability compared to baseline methods.

6 Instruction Fine-Tuning of Large Language Models

6.1 Experimental Setups

Datasets. Our instruction tuning data were derived from Super-Natural Instructions (Wang et al., 2022). We selected 17 representative tasks: 9 natural language generation (NLG) tasks and 8 natural language understanding (NLU) tasks. Following (Wang et al., 2022), we randomly designated 4 datasets as held-out test sets and used the remaining 13 datasets for training. This resulted in a training set of 1,950 examples and a test set of 400 examples (see Appendix C.1 for details). Each example follows a format of (p, x, y), where p is an ICL prompt containing 2 to 5 demonstrations.

Evaluation Metrics. Following the practice in Super-Natural Instructions (Mishra et al., 2022; Wang et al., 2022), we adopt ROUGE-L (Lin, 2004) for reporting performance results, due to the diversity of our tasks and the open-ended nature of instruction tuning. We also report a single "average" metric across all datasets, following the methodology in FLAN (Wei et al., 2022; 2023).

Baseline and Implementation Details. To evaluate our framework, we compare it with other learning algorithms, including **Empirical Risk Minimization (ERM) (Min et al., 2022)**, **ERM with Demonstration Shuffling (ERM+DS)** (Zhang et al., 2018), **ERM with Instance Mixup (ERM+IM) (Zhang et al., 2018)**, **InfoAC** (Xiang et al., 2024), **Batch-ICL** (Zhang et al., 2024) and **CurStable** (Chang & Jia, 2023). We use FLAN-large as the P-Net and experiment with five LLMs: **Llama3-8B, Llama2-7B/13B, Mistral-7B, Gemma-7B**. More details on baselines and implementations are in Appendix C.2. Hyperparameters are in Appendix C.3.

6.2 Evaluation Results

We evaluate PEARL from three perspectives: (1) comparison with training-stage methods (empirical risk minimization [ERM] and its augmentations, InfoAE), (2) comparison and integration with inference-stage techniques (Batch-ICL, CurStable), and (3) scalability to many-shot in-context learning (ICL; Agarwal et al., 2024) with more demonstrations.

Table 2 presents the comparative performance of various methods. PEARL consistently improves both average and worst-case performance across all unseen tasks. As the number of shots increases, the worst-case performance gain relative to ERM progressively increases from 14.2% at 2 shots to 29.4% at 4 shots. Notably, while optimized for worst-case performance, PEARL also achieves superior average performance with gains of 5.7-9.8%. This improvement may stem from the rapid convergence observed during LLaMA3-8B's fine-tuning, where training loss plateaus within one epoch. The rapid convergence suggests that focusing on challenging permutations during training is more effective than using random ones—an observation consistent with WizardLM (Xu et al., 2024).

PEARL outperforms all training-stage and inference-stage methods by a large margin. On inference-stage methods, Batch-ICL boosts both average and worst-case performance on classification tasks

Table 2: Average and worst-case performance of Llama3-8B on held-out tasks with varying shots. Results span four datasets: CommonsenseQA (CSQA), Curiosity Dialogue (CurDial), CoLA, and Tell Me Why (TMW). Performance gains (%) over ERM shown in blue.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
2	ERM	57.3	49.4	58.0	54.0	57.9	43.4	62.0	58.0	51.1	42.0
	ERM+DS	57.5 **(-0.2)**	48.6 **(-1.6)**	62.0	54.0	54.1	37.8	61.0	60.0	51.5	42.7
	ERM+IM	53.5 **(-6.6)**	44.4 **(-10.1)**	63.0	54.0	44.7	28.1	57.0	56.3	49.4	39.2
	InfoAC	55.7 **(-2.9)**	47.6 **(-3.7)**	57.5	56.0	53.4	36.4	63.0	61.5	48.7	37.3
	CurStable	61.6 **(+7.5)**	52.1 **(+5.4)**	64.0	56.0	61.7	46.2	68.4	62.0	52.3	44.1
	Batch-ICL	58.6 **(+2.2)**	-	63.0	-	56.3	-	65.5	-	49.6	-
	Pearl	**62.9** **(+9.8)**	**56.4** **(+14.2)**	**65.0**	**62.0**	**60.3**	**50.7**	**71.0**	**68.0**	**55.1**	**44.8**
	+ CurStable	65.6 **(+14.5)**	58.0 **(+17.4)**	68.0	63.0	64.6	52.8	74.0	70.0	55.9	46.2
	+ Batch-ICL	-	-	**65.5**	-	-	-	**72.0**	-	-	-
3	ERM	57.8	38.3	57.7	47.0	61.4	25.9	61.9	52.0	50.3	29.4
	ERM+DS	56.1 **(-2.9)**	39.7 **(+3.7)**	60.0	46.0	54.1	25.4	60.0	**56.0**	50.3	31.5
	ERM+IM	55.3 **(-4.3)**	39.8 **(+3.9)**	59.0	46.0	54.6	28.0	57.6	53.1	50.0	31.9
	InfoAC	56.3 **(-2.6)**	39.5 **(+3.1)**	59.3	49.0	55.2	24.3	62.1	55.8	48.4	28.8
	CurStable	61.0 **(+5.4)**	41.4 **(+8.0)**	65.0	52.0	62.5	26.7	64.0	54.0	52.3	32.7
	Batch-ICL	58.6 **(+1.3)**	-	62.0	-	59.6	-	64.0	-	48.7	-
	Pearl	**63.1** **(+9.2)**	**46.9** **(+22.5)**	**68.4**	**62.0**	**66.7**	**34.8**	**64.7**	**56.0**	**52.4**	**34.7**
	+ CurStable	65.0 **(+12.5)**	48.9 **(+27.5)**	70.0	64.0	67.6	35.8	68.4	58.0	54.1	37.6
	+ Batch-ICL	-	-	**68.7**	-	-	-	**65.6**	-	-	-
4	ERM	59.7	30.6	61.3	38.0	62.9	21.3	63.3	45.8	**51.1**	17.5
	ERM+DS	57.7 **(-3.4)**	31.8 **(+3.9)**	63.3	40.0	57.3	17.6	60.1	**52.0**	49.9	17.8
	ERM+IM	56.0 **(-6.2)**	32.4 **(+5.9)**	63.2	42.0	53.7	17.8	57.6	48.5	49.6	21.3
	InfoAC	58.6 **(-1.8)**	33.0 **(+7.8)**	63.7	44.0	58.7	19.0	63.9	51.0	48.1	17.0
	CurStable	60.8 **(+1.8)**	32.3 **(+5.6)**	63.0	40.0	64.5	22.8	64.1	48.0	51.5	18.4
	Batch-ICL	58.5 **(-2.0)**	-	62.0	-	61.5	-	63.3	-	47.2	-
	Pearl	**63.1** **(+5.7)**	**39.6** **(+29.4)**	**68.4**	**52.0**	**69.2**	**31.3**	**64.7**	**52.0**	50.1	**23.0**
	+ CurStable	65.0 **(+8.8)**	41.4 **(+35.1)**	70.6	54.0	72.3	34.2	66.3	54.0	50.6	23.2
	+ Batch-ICL	-	-	**69.0**	-	-	-	**65.0**	-	-	-

(CSQA, CoLA); however, it exhibits limited or negative effects on generation tasks (CurDial, TMW), limiting applicability. In contrast, CurStable performs well on both task types via demonstration selection. Moreover, combining PEARL with inference-stage methods further improves performance.

We evaluate PEARL and ERM in the many-shot ICL setting. As shown in Figure 5, PEARL achieves surprising worst-case performance gains from 24% to 40% when generalizing to larger shot numbers, despite being trained with significantly fewer shots and shorter sequences. This suggests our method helps LLMs learn robust features that generalize well to many-shot ICL. Detailed results are in Appendix F.

Analyses of hyperparameters and extended experiments on LLaMA2, Mistral, and Gemma are provided in Appendices D and E, respectively.

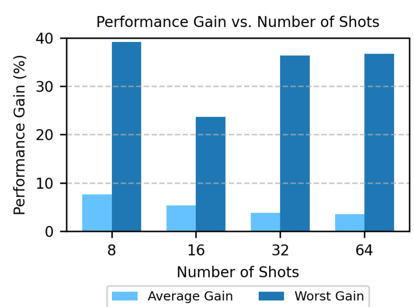

Figure 5: Scaling to many-shot ICL.

7 Conclusion

We introduced a novel permutation-resilient learning framework (PEARL) to enhance the robustness of LLMs against different permutations. PEARL employs a hard Permutation mining Network (P-Net) that utilizes the Sinkhorn algorithm to generate challenging permutations, combined with adversarial training to systematically improve LLM performance. Through empirical evaluations in both the synthetic ICL task and the instruction tuning task, our framework has proven effective in mitigating attacks and enhancing the generalization of LLMs. This research addresses a significant vulnerability in LLMs, setting a foundation for the development of more resilient future language models.

REFERENCES

Ryan Prescott Adams and Richard S. Zemel. Ranking via sinkhorn propagation, 2011.

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning, 2024. URL https://arxiv.org/abs/2404.11018.

A. Ben-Tal, D. den Hertog, A. D. Waegenaere, B. Melenberg, and G. Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59:341–357, 2013.

Aharon Ben-Tal, Dick den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Advanced Risk & Portfolio Management® Research Paper Series*, 2011. URL https://api.semanticscholar.org/CorpusID:761793.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pp. 41–48, 2009.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Ting-Yun Chang and Robin Jia. Data curation alone can stabilize in-context learning, 2023. URL https://arxiv.org/abs/2212.10378.

Yongqiang Chen, Binghui Xie, Kaiwen Zhou, Bo Han, Yatao Bian, and James Cheng. Positional information matters for invariant in-context learning: A case study of simple function classes, 2023. URL https://arxiv.org/abs/2311.18194.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. URL http://jmlr.org/papers/v24/22-1144.html.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL http://jmlr.org/papers/v25/23-0870.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019b. URL https://arxiv.org/abs/1810.04805.

J. Duchi, P. Glynn, and H. Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv*, 2016.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598. Curran Associates, Inc., 2022.

Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1608–1617. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/genevay18a.html.

Qi Guo, Leiyu Wang, Yidong Wang, Wei Ye, and Shikun Zhang. What makes a good order of examples in in-context learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14892–14904, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.884. URL https://aclanthology.org/2024.findings-acl.884.

Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, and Yu Bai. How do transformers learn in-context beyond simple functions? a case study on learning with representations. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=ikwEDva1JZ.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

W. Hu, G. Niu, I. Sato, and M. Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning (ICML)*, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Leonid Kantorovich. On the transfer of masses. In *Doklady Akademii Nauk*, volume 37, pp. 227–229, 1942.

H. Lam and E. Zhou. Quantifying input uncertainty in stochastic optimization. In *2015 Winter Simulation Conference*, 2015.

Hongjing Li, Hanqi Yan, Yanran Li, Li Qian, Yulan He, and Lin Gui. Distinguishability calibration to in-context learning. In Andreas Vlachos and Isabelle Augenstein (eds.), *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 1385–1397, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.102. URL https://aclanthology.org/2023.findings-eacl.102.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. Less is more: Pretrain a strong Siamese encoder for dense text retrieval using a weak decoder. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2780–2791, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.220. URL https://aclanthology.org/2021.emnlp-main.220.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556.

Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2791–2809, Seattle, United States, July 2022. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.244. URL https://aclanthology.org/2022.acl-long.244.

T. Miyato, S. Maeda, S. Ishii, and M. Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

Gaspard Monge. Memoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

Y. Oren, S. Sagawa, T. Hashimoto, and P. Liang. Distributionally robust language modeling. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. Revisiting demonstration selection strategies in in-context learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9090–9101, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.492. URL https://aclanthology.org/2024.acl-long.492.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380959. doi: 10.1145/3411763.3451760. URL https://doi.org/10.1145/3411763.3451760.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxGuJrFvS.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014. ISBN 1107057132.

Richard Sinkhorn. A relationship between arbitrary positive matrices and stochastic matrices. *Canadian Journal of Mathematics*, 18:303–306, 1966. doi: 10.4153/CJM-1966-033-9.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer: New York, 1999.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks, 2022. URL `https://arxiv.org/abs/2204.07705`.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Jerry Wei, Le Hou, Andrew Lampinen, Xiangning Chen, Da Huang, Yi Tay, Xinyun Chen, Yifeng Lu, Denny Zhou, Tengyu Ma, and Quoc Le. Symbol tuning improves in-context learning in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 968–979, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.61. URL `https://aclanthology.org/2023.emnlp-main.61`.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? *arXiv preprint arXiv:2012.03107*, 2020.

Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. Addressing order sensitivity of in-context demonstration examples in causal language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6467–6481, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.386. URL `https://aclanthology.org/2024.findings-acl.386`.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=CfXh93NDgH`.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018. URL `https://arxiv.org/abs/1710.09412`.

Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. Batch-ICL: Effective, efficient, and order-agnostic in-context learning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10728–10739, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.638. URL `https://aclanthology.org/2024.findings-acl.638`.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/zhao21c.html`.

Appendix

A Adversarial Training Algorithm

We present the adversarial training algorithm in Table A.

Algorithm 1 Adversarial Training Algorithm

1: **Input**: Training data $\hat{P}$, LLM θ, P-Net ϕ, P-Net iteration step m, Entropy coefficient β.
2: **repeat**
3:     **for** $t = 1, \ldots, m$ **do**
4:         Sample an example (p, x, y) from $\hat{P}$.
5:         Generate a permutation Π using P-Net: $\Pi \sim$ P-Net$(\phi; p, x, y)$.
6:         Compute the LLM loss on the permuted input $(\Pi \cdot p, x, y)$: $L(\phi; \theta)_{lm}$.
7:         Compute the entropy regularization term $L(\phi)_{ent}$
8:         Update P-Net parameters ϕ by ascending the gradient $\nabla_\phi L(\phi; \theta)_{lm} - \beta \nabla_\phi L(\phi)_{ent}$.
9:     **end for**
10:    Update LLM parameters θ by descending its gradient $\nabla_\theta L(\phi; \theta)_{lm}$.
11: **until** convergence
12: **Output**: Optimized parameters θ and ϕ.

B Detailed Setup of ICL With Linear Functions

B.1 Datasets Construction

We investigate training a language model to perform in-context learning on linear functions, following (Garg et al., 2022; Guo et al., 2024b). The function class is defined as $\mathcal{F} = \{f \mid f(x) = w^\top x, w \in \mathbb{R}^d\}$, where d is the input dimension. Each data sample is constructed as follows:

(a) Function sampling: A weight vector $w \sim \mathcal{N}(0, I_d)$ is sampled, defining a linear function $f(x) = w^\top x$.

(b) Input sampling: Inputs $x_1, x_2, \ldots, x_{k+1} \sim \mathcal{N}(0, I_d)$ are independently drawn.

(c) Output generation: For each input, the corresponding output is computed as $y_i = f(x_i) = w^\top x_i$ for $i = 1, 2, \ldots, k+1$.

The input prompt p^i consists of i demonstrations and the $(i+1)$-th example as the query: $p^i = (x_1, f(x_1), x_2, f(x_2), ..., x_i, f(x_i), x_{i+1})$. We trained a language model L_θ, parameterized by θ, to minimize the expected loss over all input prompts:

$$\min_\theta \mathbb{E}_p \left[\frac{1}{k+1} \sum\nolimits_{i=0}^{k} \ell(LM_\theta(p^i), f(x_{i+1})) \right], \tag{19}$$

where $l(\cdot)$ is the mean squared error (MSE) loss. During testing, we evaluated performance using the same MSE metric. We report the normalized squared error ($(LM(p) - w^\top x_{\text{query}})^2/d$), where d is the problem dimension.

B.2 Implement Details

Architecture. Following (Garg et al., 2022), we implement L_θ using a GPT-2 architecture (Radford et al., 2019) with 12 layers, 8 attention heads, and a hidden dimension of 256. The model takes as input a sequence of vectors in its embedding space and predicts the next vector in the sequence within the same space.

Training. We pre-train the model from scratch on a generated dataset of 40k linear functions using the AdamW (Loshchilov & Hutter, 2019). We employ a batch size of 128 and trained for 500k steps, selecting the best checkpoint based on validation set performance. In the PEARL framework, we randomly initialize the P-Net with a BERT-base-sized transformer encoder, also pre-training it from scratch. During testing, we sample novel functions to assess the model's ability to infer new weights w through in-context demonstrations.

C Detailed Setup of Instruction Fine-Tuning

C.1 Details of Datasets

Table 4: Details of datasets used in instruction tuning from natural instructions.

Task ID	Task Name	Source	Category
1297	QASC Question Answering	QASC	Question Answering
442	COM_QA Paraphrase Question Generation	COM_QA	Question Rewriting
908	DialogRE Identify Familial Relationships	DialogRE	Speaker Relation Classification
288	Gigaword Summarization	Gigaword	Title Generation
582	Natural Questions Answer Generation	Natural Questions	Question Answering
151	TOMQA Find Location Easy Clean	TOM_QA	Question Answering
1714	ConvAI3 Sentence Generation	ClariQ	Dialogue Generation
379	AGNews Topic Classification	AG News	Text Categorization
639	MultiWOZ User Utterance Generation	MultiWOZ 2.2	Dialogue Generation
209	Stance Detection Classification	StarCon	Stance Detection
1516	IMPPRES Natural Language Inference	IMPPRES	Textual Entailment
589	Amazon Food Summary Text Generation	Amazon Reviews	Summarization
1285	KPA Keypoint Matching	ArgKP	Text Matching

Table 3: Summary of datasets used in instruction tuning.

Split	Category	# Tasks	# Samples
Training	NLG	7	1050
	NLU	6	900
Testing	NLG	2	200
	NLU	2	200

Our instruction tuning data are derived from Super-Natural Instructions (Wang et al., 2022), which are part of the FLAN v2 benchmark (Chung et al., 2024). We selected 17 representative tasks, comprising 9 natural language generation (NLG) tasks and 8 natural language understanding (NLU) tasks. Following the methodology of Wang et al. (2022), we randomly designated 4 datasets as held-out test sets and used the remaining 13 datasets for training. Each training dataset contains 150 examples, and each test dataset contains 100 examples, resulting in a training set of 1,950 examples and a test set of 400 examples, as summarized in Table 3 (details are provided in the Appendix C.1. The details of datasets used in instruction tuning is presented in Table 4.

C.2 Baseline and Implementation Details

To evaluate the performance of our trained model, we compare it with other learning algorithms.

Empirical Risk Minimization (ERM) (Min et al., 2022): Standard approach minimizing the average loss over the training dataset, adopted by mainstream instruction tuning models such as FLAN (Chung et al., 2024), Natural Instructions (Mishra et al., 2022; Wang et al., 2022), and MetaICL (Min et al., 2022).

ERM with Demonstration Shuffling (ERM+DS) (Zhang et al., 2018): Enhances ERM by randomly shuffling the order of in-context demonstrations within each sample at each training step. This introduces robustness by exposing the model to different permutations of demonstrations during training. It can be considered a form of epoch-level data augmentation.

ERM with Instance Mixup (ERM+IM)(Zhang et al., 2018): Incorporates *Instance Mixup* technique during each training step. For each data point, we generate multiple augmented versions by randomly selecting different in-context demonstrations. We perform multiple forward passes to compute the loss for each augmented version, average these losses, and then perform a single backward pass using the averaged loss. This approach provides finer-grained data augmentation compared to demonstration shuffling. Notably, by comparing this baseline with our method, we contrast min-mean optimization (ERM+IM) with min-max optimization (our method).

InfoAC: (Xiang et al., 2024) is a training-stage method that employs contrastive learning to enable earlier tokens to access information from later tokens, amining to mitigate the order sensitivity of ICL inherent in autoregressive LM.

Batch-ICL: (Zhang et al., 2024) is an inference-stage method that transforms an n-shot ICL prompt into n individual one-shot prompts and then ensembles the results to improve robustness.

CurStable: (Chang & Jia, 2023) is another inference-stage method that enhances ICL performance by selecting optimal demonstration samples. This selection process involves performing multiple inferences with different prompts on a validation set, calculating the expected performance when

Category	Hyperparameter	Value
LLMs	Learning rate	3e-5
	Batch size	16
	Max sequence length	512
	Weight decay coefficient	0.1
	Epoch	2
LoRA	Rank	8
	Alpha	32
	Dropout	0.1
	P-Net target modules	q, v
	LLMs target modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
P-Net	Temperature	0.1
	Iteration coefficient	80
	Entropy constraint	1.0
	Noise	0.3
	Learning rate	1e-4
	Batch size	16
	Max sequence length	512

Table 5: Hyperparameter settings used in our main experiment.

each demonstration is used, and assigning an importance score to each. The demonstrations with the highest scores are then selected to form the demonstration pool.

By including these baselines—training-stage (ERM, ERM+DS, ERM+IM, and InfoAC) and inference-stage (Batch-ICL and CurStable)—we provide a comprehensive evaluation of our proposed method.

As for the proposed PEARL framework, we select the LLaMA3-8B model as our LLM and the FLAN-large encoder as the P-Net. Both models are fine-tuned using LoRA (Hu et al., 2022), with the number of finetuned parameters of P-Net being 1/20 that of the LLM. We train the models on the instruction dataset for two epochs using a single NVIDIA A40 GPU, with a batch size of 16, resulting in a total of 246 training steps. The optimizer used was AdamW. The learning rates for the P-Net and the LLM are set to 1×10^{-4} and 3×10^{-4}, respectively. For the Sinkhorn algorithm, we use 80 iterations, a temperature parameter of 0.1, and an entropy constraint coefficient $\beta = 1.0$.

C.3 Details of Hyperparameter Settings

In this section, we provide a comprehensive overview of the hyperparameter settings used in our experiments (Table 5). The hyperparameters can be categorized into three groups: (1) basic LLM training parameters, such as learning rate and batch size; (2) LoRA configuration parameters; and (3) P-Net optimization parameters. These hyperparameters were selected based on average validation performance and kept consistent across comparative experiments to ensure fair comparison.

D Analysis of Hyperparameters In Instruction Finetuning

We conduct analysis to understand the impact of key hyperparameters on P-Net learning and our overall framework. Our analysis focuses on two main aspects: the effect of the entropy constraint strength, and the influence of iteration number and temperature in the Sinkhorn algorithm.

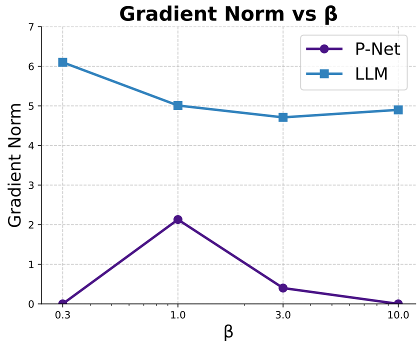

Figure 6: Impact of entropy coefficient.

Influence of Entropy Regularization in OT We examine the impact of the entropy regularization coefficient in OT, testing values of 0.3, 1.0, 3.0, and 10.0 (Figure 6). At a low coefficient (0.3), P-Net's gradient norm remained small, indicating minimal learning and potential generation of simple semantic overlaps to satisfy adversarial training requirements. Concurrently, the LLM's gradient norm struggled to decrease. The gradient norm for P-Net peaked at 1.0, suggesting optimal learning conditions. As coefficients increased to 3.0 and 10.0, P-Net's gradient norm decreased again, suggesting excessive restrictions. The range of 1.0-3.0 provided an ideal balance, encouraging P-Net to extract meaningful information from the LLM without oversimplifying or overcomplicating the task. In contrast, the LLM's gradient norm decreased consistently with increasing coefficients, indicating a distinct response to entropy regularization.

Table 6: Impact of number of iterations and temperature on the average/worst-case performance.

# Iter.	Temperature		
	0.03	0.1	0.3
80	55.7 / 40.0	55.7 / 40.0	55.4 / 39.6
200	55.7 / 40.0	55.8 / 40.0	55.8 / 40.6

Effect of Sinkhorn Algorithm Parameters We investigate the interplay between two critical parameters in the Sinkhorn algorithm: number of iterations and temperature. Intuitively, these parameters are positively correlated; higher iteration counts typically allow for higher temperatures. Our experiments, however, reveal an unexpected robustness to parameter variations. With the entropy regularization coefficient fixed at 1, we vary the number of iterations $(80, 200)$ and temperature $(0.03, 0.1, 0.3)$. As presented in Table 6, surprisingly, these substantial parameter changes result in minimal performance variation. This suggests that the Sinkhorn algorithm in our framework is less sensitive to these parameters than initially hypothesized, potentially indicating a wider range of stable configurations for practical applications.

E Extended Instruction Finetuning Across Diverse LLMs

We expanded our evaluation to include additional models: Mistral-7B, Gemma-7B, and earlier generations such as Llama2-7B and Llama2-13B, as detailed in the tables from Table (7) to Table (9).

Sensitivity to Permutations Across LLM Families Our analysis reveals that different LLM families exhibit varying degrees of sensitivity to permutations. The sensitivity ranking, from highest to lowest, is as follows: Llama, Gemma, and Mistral. Notably, all examined families showed significant performance declines, typically exceeding 10 percentage points.

Adaptation of the Proposed Method In scenarios with three or more examples, our method consistently demonstrated substantial improvements, often enhancing worst-case performance by more than 10%. These results confirm the robustness and effectiveness of our approach.

F Scaling to Many-Shot In-Context Learning

Table 7: Instruction fine-tuning results for Mistral-7B evaluated on four held-out tasks. Performance gains (%) over the ERM baseline are indicated in blue.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
2	ERM	64.1	58.1	67.0	64.0	54.6	41.8	81.0	78.0	53.7	48.5
	PEARL	67.0 **(+4.5)**	62.4 **(+7.5)**	68.0	66.0	59.4	49.0	82.0	78.0	58.4	56.7
3	ERM	66.6	56.1	67.0	62.0	63.7	38.9	80.0	76.0	55.6	47.3
	PEARL	69.5 **(+4.3)**	62.8 **(+12.0)**	70.0	66.0	70.1	60.1	83.6	78.0	54.1	47.0
4	ERM	66.7	50.4	68.9	60.0	67.6	47.8	74.2	52.0	55.9	41.6
	PEARL	68.3 **(+2.5)**	57.1 **(+13.4)**	69.9	62.0	71.6	54.8	74.9	66.0	56.8	45.5
5	ERM	67.9	50.7	67.5	56.0	70.7	52.6	76.0	56.0	57.4	38.2
	PEARL	70.2 **(+3.4)**	58.1 **(+14.5)**	70.4	64.0	76.7	59.3	73.3	66.0	60.4	43.0

Table 8: Instruction fine-tuning results for Gemma-7B evaluated on four held-out tasks. Performance gains (%) over the ERM baseline are indicated in blue.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
2	ERM	66.2 **(+0.0)**	59.5 **(+2.0)**	71.0	70.0	59.1	46.1	77.0	70.0	57.8	52.0
	PEARL	66.3	60.7	74.0	68.0	47.3	39.2	82.0	78.0	61.7	57.6
3	ERM	64.7 **(+5.8)**	52.5 **(+13.0)**	70.7	64.0	67.1	45.2	70.3	60.0	50.5	40.7
	PEARL	68.4	59.3	74.7	68.0	59.2	42.5	78.7	76.0	61.0	50.6
4	ERM	65.0 **(+3.4)**	46.5 **(+13.0)**	65.0	54.0	71.4	41.1	72.5	58.0	51.1	32.9
	PEARL	67.2	52.5	71.4	60.0	60.7	38.9	75.9	66.0	60.8	45.2
5	ERM	64.3 **(+3.1)**	46.3 **(+10.2)**	65.9	54.0	73.4	48.3	65.6	50.0	52.3	32.9
	PEARL	66.3	51.0	70.3	60.0	63.4	43.6	71.3	60.0	60.2	40.4

We evaluate the scalability of PEARL by extending our analysis to many-shot scenarios, testing performance with 8 to 64 in-context examples (Table 11). Notably, despite being trained solely on 5-shot demonstrations, PEARL exhibits strong generalization to settings with substantially more examples. Using Llama3-8B as our base model, we compare PEARL and ERM training approaches across four held-out tasks. Our analysis reveals persistent performance advantages of PEARL over the ERM baseline across all shot regimes.

G Best-case Performance

Although our methodology was initially designed to optimize for pessimistic (worst-case) scenarios, we have also included an evaluation of the best-case performance for both PEARL and ERM to provide a balanced perspective. The results are shown in the Table **??**.

Surprisingly, the results, show that across all datasets and in every shot condition, PEARL's best performance consistently exceeded that of ERM, although the overall average improvement was modest.

Table 9: Instruction fine-tuning results for Llama2-7B evaluated on four held-out tasks. Performance gains (%) over the ERM baseline are indicated in blue.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
2	ERM	56.6 **(+1.5)**	46.3 **(+0.4)**	56.0	50.0	61.3	50.2	58.2	42.0	50.7	43.1
	PEARL	57.4	46.5	58.0	48.0	55.2	44.7	62.0	48.0	54.4	45.4
3	ERM	58.2 **(+2.3)**	34.0 **(+19.1)**	52.7	34.0	64.0	36.4	66.0	36.0	50.1	29.4
	PEARL	59.6	40.4	56.3	40.0	66.2	46.2	67.0	42.0	48.7	33.5
4	ERM	58.9 **(+2.7)**	19.9 **(+59.1)**	60.0	26.0	68.1	24.4	60.2	14.0	47.3	15.1
	PEARL	60.5	31.6	61.2	40.0	69.4	40.1	62.4	24.0	48.9	22.4
5	ERM	61.9 **(+1.6)**	25.8 **(+24.7)**	59.0	32.0	74.2	43.9	65.7	10.0	48.6	17.1
	PEARL	62.9	32.1	62.4	38.0	73.3	43.4	64.8	24.0	51.0	23.0

Table 10: Instruction fine-tuning results for Llama2-13B evaluated on four held-out tasks. Performance gains (%) over the ERM baseline are indicated in blue.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
2.0	ERM	66.3 **(+2.4)**	56.6 **(+7.3)**	56.0	46.0	72.6	56.2	83.0	76.0	53.4	48.0
	PEARL	67.9	60.7	64.0	58.0	73.8	64.2	81.0	76.0	52.6	44.4
3.0	ERM	65.7 **(+4.2)**	46.2 **(+8.7)**	55.7	38.0	76.4	51.3	77.7	56.0	53.1	39.6
	PEARL	68.5	50.3	62.7	44.0	81.0	58.4	76.7	56.0	53.5	42.6
4.0	ERM	65.8 **(+0.9)**	33.2 **(+21.1)**	58.2	28.0	79.6	41.6	73.7	38.0	51.8	25.0
	PEARL	66.4	40.2	63.3	42.0	80.4	45.5	69.4	42.0	53.1	29.1

Table 11: Performance evaluation across 8-, 16-, 32-, and 64-shot settings comparing PEARL and ERM learning algorithm for Llama3-8B on four held-out tasks, with gains (%) relative to the ERM.

		Average		CSQA		CurDial		CoLA		TMW	
# Shot	Method	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.	Avg.	Worst.
8	ERM	61.8 **(+7.6)**	21.3 **(+39.2)**	61.4	36.0	68.3	22.7	62.7	16.0	54.8	10.6
	PEARL	66.5	29.7	67.7	44.0	77.1	28.7	65.0	32.0	56.2	14.0
16	ERM	66.9 **(+5.3)**	21.3 **(+23.7)**	67.3	36.0	76.5	31.4	67.2	8.0	56.5	9.7
	PEARL	70.5	26.3	70.9	46.0	83.9	37.5	70.1	12.0	56.9	9.8
32	ERM	67.4 **(+3.8)**	19.3 **(+36.4)**	67.5	32.0	77.8	30.7	68.2	6.0	56.1	8.6
	PEARL	70.0	26.4	70.0	44.0	82.6	40.3	70.6	12.0	56.6	9.1
64	ERM	68.1 **(+3.5)**	20.6 **(+36.7)**	68.1	38.0	76.9	27.7	72.2	8.7	55.0	8.0
	PEARL	70.4	28.2	69.5	46.0	82.9	38.9	74.2	19.6	55.1	8.1

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