Mixtures of In-Context Learners

Giwon Hong[1] **Emile van Krieken**[1] **Edoardo M. Ponti**[1]
Nikolay Malkin[1] **Pasquale Minervini**[1,2]
[1]University of Edinburgh, United Kingdom [2]Miniml.AI, United Kingdom
{giwon.hong, p.minervini}@ed.ac.uk

Abstract

In-context learning (ICL) adapts LLMs by providing demonstrations without fine-tuning the model parameters; however, it is very sensitive to the choice of in-context demonstrations, and processing many demonstrations can be computationally demanding. We propose Mixtures of In-Context Learners (MoICL), a novel approach that uses subsets of demonstrations to train a set of experts via ICL and learns a weighting function to merge their output distributions via gradient-based optimisation. In our experiments, we show performance improvements on 5 out of 7 classification datasets compared to a set of strong baselines (e.g., up to +13% compared to ICL and LENS). Moreover, we improve the Pareto frontier of ICL by reducing the inference time needed to achieve the same performance with fewer demonstrations. Finally, MoICL is more robust to out-of-domain (up to +11%), imbalanced (up to +49%) and perturbed demonstrations (up to +38%).[1]

1 Introduction

In-context learning (ICL) refers to the ability of a large language model (LLM) to learn to perform tasks by interpreting and adapting to examples (demonstrations) provided directly in the input context without requiring updates to its parameters (Brown et al., 2020; Wei et al., 2022). However, in ICL, the number of demonstrations may be limited by the memory requirements of the model (Wei et al., 2022), and its effectiveness can vary significantly depending on which demonstrations are selected (Lu et al., 2022; Chen et al., 2023b) and how they are verbalised (Voronov et al., 2024). Current methods for selecting demonstrations are largely heuristic and do not adequately quantify the influence of individual examples on the generalisation properties of the model (Lu et al., 2024). Demonstrations are often selected randomly or based on simple heuristics (Xu et al., 2024), which can lead to suboptimal results.

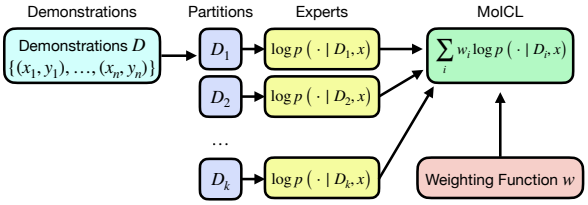


Figure 1: A Mixture of In-Context Learners (MoICL) first partitions a set of demonstrations D in k partitions to create k experts trained via in-context learning, and then combines their next-token predictions via a trainable weighting function.

To address these issues, we propose Mixtures of In-Context Learners (MoICL), a method for dynamically learning how different sets of examples contribute to the prediction task. MoICL prompts an LLM with multiple subsets of examples and combines their next-token distributions via a weighting function that can be trained via gradient-based optimisation methods; Fig. 1 shows a high-level outline of the method.

We analyse the generalisation properties of MoICL in the following settings: (1) presence of out-of-domain (OOD) demonstrations, where some in-context demonstrations are sourced from a different dataset; (2) label imbalance, where the training label distribution is significantly skewed towards a subset of labels; and (3) perturbed demonstrations, where the labels of some demonstrations are perturbed to be completely incorrect. In all cases, we find that MoICL produces significantly more accurate results than ICL.

Furthermore, MoICL does not require access to the internal parameters of the LLM, making it applicable to black-box LLMs, and it significantly reduces the computational complexity issues arising from long contexts since it allows the distribution of the training samples among multiple experts. We also show that MoICL can be made more efficient

[1]Our code is available at https://github.com/HongGiwon/Mixture-of-In-context-Learners

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 26332–26351

by sparsifying the mixing weights.

We summarise our contributions as follows:

- We introduce Mixtures of In-Context Learners (MoICL), a method for combining the predictions of multiple models trained via ICL and identifying the optimal mixture weights via gradient-based optimisation. (§2)
- We show that MoICL is resilient to perturbed demonstrations and label imbalance. (§3)
- We demonstrate that MoICL is competitive with standard ICL while being significantly more data-, memory-, and compute-efficient. (§4)

2 Mixtures of In-Context Learners

2.1 In-Context Learning

In ICL, given a LLM with next-token distribution $p(\cdot)$, a set of n demonstrations $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and an input instance x, the model generates a response y when prompted with the concatenation of the examples in D and x:

$$\begin{aligned} y &\sim p(y \mid x_1, y_1, \ldots, x_n, y_n, x) \\ &= p(y \mid D, x), \end{aligned} \tag{1}$$

we refer to the model in Eq. (1) as *concat-based ICL* (Min et al., 2022a). With concat-based ICL, given a demonstration set D, the model can generate a response y for the input text x without needing task-specific fine-tuning or access to the model parameters. However, concat-based ICL is still problematic; recent works show that it is very sensitive to the choice of the prompts and in-context demonstrations (Voronov et al., 2024), the number of demonstrations is bounded by the maximum context size (Brown et al., 2020), and, in Transformer-based LLMs, the cost of self-attention operations grows quadratically with the number of in-context samples (Liu et al., 2022).

2.2 Mixtures of In-Context Learners

We propose Mixtures of In-Context Learners (MoICL), a method for addressing the limitations of concat-based ICL (§2.1). We first partition the set of demonstrations D into k disjoint subsets $D_1, \ldots, D_k$, i.e. $D = D_1 \sqcup \ldots \sqcup D_k$.[2] Then, each demonstration subset $D_i \subseteq D$ is passed to the LLM along with the input text x, and we denote the LLM predictors with the different demonstrations in their context as *experts*. The next-token distributions of the experts are combined using a vector of mixing weights $w \in \mathbb{R}^k$:

$$p(y \mid D, x) \propto \exp\left[\sum_{i=1}^{k} w_i \log p(y \mid D_i, x)\right] \tag{2}$$

where each $w_i \in \mathbb{R}$ represents the contribution of the expert denoted by $p(y \mid D_i, x)$ to the final next-token distribution $p(y \mid D, x)$, and each $p(y \mid D_i, x)$ is a concat-based ICL expert, as in Eq. (1), with its own set of demonstrations. [3]

For computing the weights $w \in \mathbb{R}^k$ of the k experts in MoICL, we use a vector of trainable parameters $w \in \mathbb{R}^k$, where w_i denotes the weight associated to the i-th expert. We refer to this parameterisation of the weighting function as *scalar weights* (*scalar*).

Adaptive Mixture Weights In addition to a vector of parameters $w \in \mathbb{R}^k$, for computing the mixture weights in MoICL, we also experiment with using a hyper-network (Ha et al., 2017) $h_\phi(\cdot)$ with parameters ϕ to generate the weights of each expert w_i, given all in-context demonstration subsets concatenated, i.e. $w_1, \ldots, w_k = h_\phi(D_1, \ldots, D_k)$. We learn the parameters of the weighting function w by maximising the conditional log-likelihood of a training set D_T. One advantage of using a hyper-network h_ϕ for dynamically computing the weights w is that the model can provide weights for sets of demonstrations D_i not seen during training. We refer to this using of a hyper-network to generate weights as *adaptive weights* (*adaptive*)

Sparsifying the Mixture Weights One limitation of MoICL is that, for each token, it requires invoking the base LLM k times, one for each expert with a different set of in-context examples. To solve this issue, we propose to *sparsify* the weighting coefficients $w \in \mathbb{R}^k$ so that only $k' < k$ of them have non-zero values. To achieve this, we define the output of the weighting function as $w = w' \odot \text{top-}k'(m)$, where $w' \in \mathbb{R}^k$ are scalar weights for the k experts, $m \in \mathbb{R}^k$ is a set of *masking coefficients*, top-k' : $\mathbb{R}^k \mapsto \{0, 1\}^k$ is a function that produces a mask that selects the highest k' elements of a k-dimensional input vector, and $\odot$ is the element-wise product. To back-propagate through the rationale extraction process, we use Implicit Maximum Likelihood Estimation (IMLE;

[2] We compare different partitioning strategies in Appendix B.2

[3] The formulation in Eq. (2) uses a product of experts; it is also possible to use a regular mixture of experts — we experimentally compare them in Fig. 2 and Appendix B.3.

Niepert et al., 2021; Minervini et al., 2023), a gradient estimation method for back-propagating through continuous-discrete functions like top-k' into neural architectures.

3 Experiments

Datasets To study how well MoICL performs on classification tasks, we use the TweetEval (Barbieri et al., 2020) offensive/hate, SST2 (Socher et al., 2013), RTE (Bentivogli et al., 2009), FEVER (Thorne et al., 2018), PAWS (Zhang et al., 2019), and QNLI (Wang et al., 2018) datasets. We report the performance on the development set for SST2, RTE, FEVER, and QNLI. For generation tasks, we use Natural Questions (NQ; Kwiatkowski et al., 2019) with an open-book setting (Lee et al., 2019), GSM8K (Cobbe et al., 2021) employing Chain-of-Thought (CoT) reasoning. For the preference fine-tuning (Appendix B.7), we use HH-RLHF (Bai et al., 2022).

Models We primarily used Llama-3-8B and its instruction-tuned model, Llama-3-8B-Instruct (AI@Meta, 2024) as our base LLMs. We use Llama-3-8B-Instruct for classification tasks, GSM8K, and HH-RLHF, while Llama-3-8B is used for NQ. In §3.4 (across multiple models), we used Llama-3.2-1B, Llama-3.2-1B-Instruct, and Llama-3.2-3B-Instruct. We use Llama-2-7b-chat, 13b-chat, and 70b-chat (Touvron et al., 2023) for analysing the influence of model scale (Appendix B.1). For longer context models in Appendix B.8, we use Llama-3.1-8B and Llama-3.1-8B-Instruct (Dubey et al., 2024). For hypernetworks, we used the T5 models (T5-efficient-tiny/mini, t5-small, t5-base; Raffel et al., 2020)

Baselines We compare MoICL with the following baselines. **Concat-based ICL** refers to the standard ICL introduced in §2.1 where all demonstrations are concatenated into a single sequence and passed as input to the LLM along with the input text. **Similarity-based Search** selects demonstrations from the demonstration pool according to their BM-25 similarity to the input text, concatenates them, and applies them in the same manner as Concat-based ICL. **Random Search** samples random subsets from the demonstration pool, concatenates them and utilizes them as in Concat-based ICL. Specifically, we sample k random subsets and select the one that performs best on the training set. Here, k is the maximum number of subsets used in MoICL, and the size of each subset is a random number between 1 and the number of demonstrations n. After finding the best subset, we evaluate it on the test set. Inspired by Wang et al. (2022), we also implement **Plurality Voting** baseline, which aggregates outputs through plurality voting over multiple inferences with different demonstrations. **Ensemble-based ICL** (Min et al., 2022a), Le et al. (2022), LARA (Huang et al., 2025), and **LENS** (Li and Qiu, 2023) were adjusted in terms of tasks and models to fit our experimental setup. We also report the results of fine-tuning the target model using LoRA (Hu et al., 2022), which requires access to the model weights. Finally, we study **MoICL Uniform** (*uniform*), an ablation that weights all experts equally, i.e. $\forall i : w_i = 1/k$. More implementation details of the baselines can be found in Appendix A.3.

Evaluation Metrics For classification tasks, we use accuracy as the evaluation metric. We use EM (Exact Match) for NQ-open and accuracy for GSM8K. For preference fine-tuning (Appendix B.7), we measure accuracy using the log probabilities of chosen and rejected responses.

More detailed settings, including hyperparameters (Appendix A.2) and implementation details with the training process (Appendix A.3), are provided in Appendix A. Furthermore, in Appendix B.2, we show that our method is not significantly affected by the choice of partitioning methods. Therefore, we applied static partitioning in all experiments.

Research Questions In our experiments, we aim to answer the following questions: (1) Does MoICL demonstrate general performance improvements over concat-based ICL and other baselines? (§3.1 and §3.2) (2) Does enforcing the mixture-weights to be non-negative improve the downstream accuracy of MoICL? (§3.3) (3) Can MoICL be extended across multiple language models? (§3.4) (4) Is MoICL resilient to problem settings involving label imbalance and perturbation? (§3.5 to §3.7) (5) Can we select demonstrations (experts) based on the tuned weights? (§3.8) (6) Can MoICL handle demonstrations that were not seen during fine-tuning? (§3.9) (7) Is MoICL more data, time, and memory-efficient compared to traditional concat-based ICL? (§4)

Method (n = 30) ↓ **Dataset** →	**Offensive**	**Hate**	**SST2**	**RTE**	**FEVER**	**PAWS**	**QNLI**
Concat-based ICL	76.44±2.48	53.54±4.29	95.46±0.14	86.43±1.26	80.63±0.49	78.12±0.77	89.08±0.44
Similarity-based Search	76.58±0.12	57.02±0.02	94.27±0.16	82.24±0.48	81.62±0.11	77.33±0.10	88.60±0.10
Random Search	77.88±1.14	58.09±1.93	**95.76±0.18**	86.57±1.43	82.13±0.10	78.88±0.57	89.99±0.26
Plurality Voting	73.21±0.60	58.51±0.78	94.06±0.20	75.38±2.18	79.42±0.42	64.97±1.02	88.98±0.17
Ensemble-based ICL (Min et al., 2022a)	73.35±0.44	53.68±4.27	95.48±0.12	86.43±1.34	80.63±0.46	65.27±0.48	88.57±0.21
Le et al. (2022)	73.14±0.47	58.63±0.62	94.38±0.19	76.32±1.89	79.40±0.43	65.29±0.50	88.54±0.23
LARA (Huang et al., 2025)	74.28±0.44	49.72±2.06	94.01±0.51	79.78±1.46	78.84±1.62	73.04±0.62	87.36±0.94
LENS (Li and Qiu, 2023)	78.70±0.67	53.20±3.11	93.81±0.16	84.98±0.74	80.07±0.29	75.60±0.72	89.04±0.40
PEFT (LoRA, Hu et al., 2022)	*79.79±4.07*	53.76±4.98	85.89±6.32	**88.88±2.78**	59.78±0.62	54.82±3.08	57.24±4.77
Mixture of ICL (*uniform*)							
$k = 5$	73.77±1.60	59.29±1.23	95.39±0.30	83.10±1.28	80.12±0.64	75.37±0.53	89.65±0.22
$k = 10$	74.00±0.87	61.70±1.61	94.91±0.19	79.93±0.81	77.47±0.89	73.49±0.46	89.65±0.14
$k = 30$	73.37±0.34	59.12±0.47	94.17±0.21	77.26±1.02	79.46±0.36	65.29±0.51	88.66±0.25
Mixture of ICL (*scalar*)							
$k = 5$	78.35±1.49	*66.03±3.31*	95.46±0.35	84.12±1.07	81.43±0.90	77.56±0.53	89.99±0.44
$k = 10$	79.42±1.48	**66.52±2.62**	95.32±0.27	83.32±1.60	*82.04±0.98*	*79.42±0.79*	**90.44±0.27**
$k = 30$	**81.33±0.69**	63.45±1.69	94.79±0.34	79.93±0.93	**82.66±0.38**	**79.50±0.33**	90.11±0.20

Table 1: Comparison between baseline methods and the proposed Mixture of In-Context Learners across classification tasks using Llama-3-8B-Instruct. n and k refer to the number of demonstrations and the number of subsets. Bold text signifies the highest accuracy for each task. For baseline methods, k is set to 30 when applicable.

3.1 MoICL in Classification Tasks

To determine the effectiveness of MoICL across various datasets, we compare it with baseline methods in Table 1. In this experiment, we set the total number of demonstrations (n) as 30 and the number of subsets (k) as 5, 10, and 30. MoICL outperformed the Baseline ICL on the Offensive, Hate, FEVER, PAWS, and QNLI datasets. The exceptions are SST2 and RTE, where MoICL performs similarly to concat-based ICL in SST2 and shows lower performance in RTE. The exception in RTE may be attributed to the inherent characteristics of the dataset; as noted in prior works (Dodge et al., 2020; Mosbach et al., 2021; Du and Nguyen, 2023), RTE is known to be particularly unstable, which may be further exacerbated in the context of initial demonstration selection. Surprisingly, MoICL *scalar* achieved the highest performance with k=10 (e.g. in Hate MoICL achieves 66.52, which is about 10 points increase compared to the concat-based ICl) or k=30 (e.g. in Offensive MoICL achieves 81.33), rather than k=5, in all tasks except for SST2 and RTE. Considering that a larger k reduces the context length (which will be further discussed in §4), MoICL manages to capture both efficiency and effectiveness.

3.2 Impact of Partitioning Size

In Fig. 2, we present the performance changes on the test set of TweetEval offensive when varying the number of subsets, k. Since the total number of demonstrations is fixed at 30, each subset contains $30/k$ demonstrations, which corresponds to the x-axis of the Figure. Note that when the number of demonstrations per subset is 30 ($k = 1$), it corresponds to the standard Concat-based ICL. We observe that *Uniform Weights* and *scalar* exhibit distinctly different patterns. With *Uniform Weights*, as the number of demonstrations per subset decreases, performance tends to decline, which is an expected outcome for ICL. However, with *scalar*, performance surprisingly increases. This seems to be because the decrease in the number of demonstrations per subset is outweighed by the increased flexibility afforded by having more subsets, each assigned tuned weights by *scalar*.

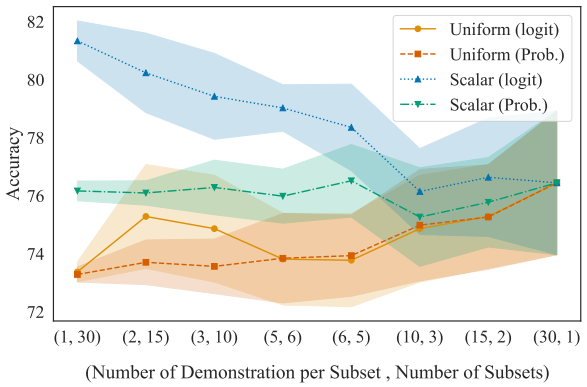


Figure 2: Test accuracy according to the number of demonstrations per subset on TweetEval Offensive dataset. The shaded area represents one standard deviation. We also compare mixing logits to mixing probabilities; see Appendix B.3.

MoICL Method ($n = k = 30$)	Accuracy
uniform	76.44±2.48
scalar ($w \in \mathbb{R}$)	**81.33±0.69**
scalar ($w \in \mathbb{R}_+$) (ReLU)	76.05±0.55
scalar ($w \in \mathbb{R}_+$) (Softplus)	73.70±0.60
scalar ($w \in \mathbb{R}_+$) (Positive Experts)	74.47±0.68

Table 2: How important is it to be able to detect anti-experts? Results on the TweetEval Offensive Test set using Llama-3-8B-Instruct. "*scalar* $w \in \mathbb{R}_+$" constraints the weights w to non-negative values. n and k refer to the number of demonstrations and the number of subsets.

3.3 Impact of Non-Negative Mixture Weights

In Mixtures of Experts, an *anti-expert* is a model that performs poorly on some inputs. Inspired by Liu et al. (2024), in MoICL, we assume that each expert could also serve as an anti-expert by allowing the expert weights to be negative; if the weight $w_i \in \mathbb{R}$ is negative, this indicates that the corresponding expert is actively being used as an anti-expert in generating the response. To assess the utility of anti-experts in MoICL, we conducted an ablation study in which expert weights were constrained to be non-negative. Specifically, we applied a ReLU activation function (Agarap, 2018) and a Softplus function (to constrain weights in a more gradient-friendly manner) during training to ensure that all weights remained non-negative, and the same constraint was enforced at inference time. We also implement **Positive Experts** method that uses weights learned from the standard *scalar* setting but ignores any expert with a negative weight during prediction.

As shown in Table 2, this restriction significantly degraded performance (81.33 → 76.05 with ReLU), suggesting that anti-experts play a critical role in improving prediction quality. Among the non-negative methods, using ReLU led to better performance than Softplus, indicating that enforcing a hard zero cutoff is more effective than a smooth constraint. Furthermore, the **Positive Experts** strategy also resulted in a notable performance drop. These findings collectively highlight the importance of allowing the model to leverage both positively and negatively contributing experts. By doing so, MoICL can not only promote useful subsets but also actively suppress useless or misleading ones, ultimately leading to more robust and accurate predictions.

Method ($n = 6$)	GSM8K (Acc.)
Llama-3.2-3B	24.76±3.12
Llama-3.2-1B-Instruct	26.47±4.27
Llama-3.2-1B	6.34±0.91
Multiple models	
Proxy-Tuning (Liu et al., 2024)	33.63±3.99
MoICL (*uniform*)	24.66±1.12
MoICL (*scalar*)	28.93±1.62
MoICL (*adaptive*)	**35.65±1.45**

Table 3: Comparison of Concat-based ICL, Proxy-Tuning, and MoICL with multiple models (§3.4) on the GSM8K dataset using Llama-3.2-1B, Llama-3.2-1B-Instruct, and Llama-3.2-3B-Instruct. k represents the number of demonstrations subset, where the total number of demonstrations is n.

Method ($n = k = 30$)	$p = 0.0$	$p = 0.5$	$p = 0.7$
Concat-based ICL	76.44±2.48	70.67±5.06	68.49±4.34
Mixture of ICL			
uniform	73.37±0.34	72.07±0.38	70.79±0.56
scalar	**81.33±0.69**	**80.95±0.65**	**80.19±0.37**

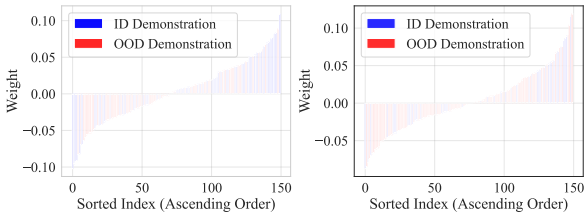
Table 4: Analysis of out-of-domain (OOD) demonstrations on TweetEval offensive test set using Llama-3-8B-Instruct. Here, p represents the proportion of OOD demonstrations sampled from the SST2 dataset. n and k refer to the number of demonstrations and subsets.

3.4 MoICL with Multiple Models

For efficiency reasons, the experts in MoICL (§2.2) use the same base model; however, this is not a strict requirement, and experts can also use different base models.[4] To analyse the impact of using MoICL with several models, we draw inspiration by Proxy-Tuning (Liu et al., 2024) and consider a MoICL composed of three models: two smaller models (Llama-3.2-1B and Llama-3.2-1B-Instruct, respectively), and a larger base model (Llama-3.2-3B) and evaluate it on GSM8K. Among the baselines, we also consider the recently proposed Proxy-Tuning (Liu et al., 2024), which merges the next-token distribution of multiple models.

Results are outlined in Table 3. MoICL with the three models led to an improvement of approximately 9.18% from concat-based ICL and 1.3% from Proxy-Tuning. Additionally, an examination of the learned weights for each model revealed values of [0.52±0.02, -0.21±0.03, 0.69±0.02] for the large base model, base model, and instruction-tuned model, respectively—aligning with the insights and findings from Proxy-tuning. This demonstrates that MoICL is extensible to multiple models and serves as a generalization of Proxy-Tuning.

[4]When vocabularies do not match, techniques like Twist Decoding (Kasai et al., 2022) can be used.



(a) 50% OOD (b) 70% OOD

Figure 3: Visualisation of the tuned weights when (a) 50% and (b) 70% of demonstrations are OOD. The y-axis indicates the weights, whereas the x-axis represents the index of demonstrations sorted in ascending order (across five different seeds). Blue bars correspond to in-domain (ID) demonstrations, and red bars correspond to out-of-domain (OOD) demonstrations.

3.5 Handling Out-of-domain Demonstrations

By learning to associate a weight to each expert, MoICL can be used to identify whether demonstrations are relevant to the task. To analyse this, in Table 4, we present the accuracy of MoICL on the TweetEval offensive test set, using a mix of demonstrations sampled from the SST dataset and those from the TweetEval offensive dataset. We observe that as p (the proportion of OOD demonstrations) increases, the performance of standard ICL methods decreases. However, MoICL (with *scalar*) effectively mitigates this by reducing the influence of these OOD demonstrations, resulting in the smallest performance drop.

This becomes even more apparent when analysing the weights of actual OOD demonstrations. When $p = 0.5$ (i.e. the number of OOD and in-domain demonstrations is equal), the average weight of in-domain demonstrations is 0.0108$\pm$0.0025, while the average weight for OOD demonstrations is -0.0059$\pm$0.0027. For $p = 0.7$, the average weight of in-domain demonstrations is 0.0127$\pm$0.0052, while the average weight for OOD demonstrations is -0.0019$\pm$0.0016. In Fig. 3, we show how the weights of in-domain demonstrations (blue bars) and OOD demonstrations (red bars) are distributed. We can see that in-domain demonstrations typically receive positive weights, while OOD demonstrations tend to receive negative weights. This provides evidence that MoICL successfully mitigates the influence of OOD demonstrations.

3.6 Mitigating Label Imbalance

To determine whether our proposed method can handle label imbalance, on the TweetEval Offensive dataset, we set up 29 "offensive" label demonstrations and one 'non-offensive' label demonstration out of 30 demonstrations. Since the TweetEval Offensive dataset has a "non-offensive" to "offensive" label ratio of 7:3, such imbalanced demonstrations would be detrimental to performance. As seen in Table 5, such imbalanced demonstrations caused a significant performance drop in standard ICL methods. However, MoICL (*scalar*) showed the least performance drop by mitigating the effects of label imbalance.

Method ($n = k = 30$)	**Original**	**Imbalanced**
Concat-based ICL	76.44$\pm$2.48	28.49$\pm$0.86
Mixture of ICL		
uniform	73.37$\pm$0.34	40.19$\pm$2.32
scalar	**81.33**$\pm$**0.69**	**77.77**$\pm$**1.20**

Table 5: Analysis of imbalanced demonstrations on the TweetEval Offensive Test set using Llama-3-8B-Instruct. "Imbalanced" refers to a condition where only one out of 30 demonstrations has a "neutral" label, while the rest are "offensive".

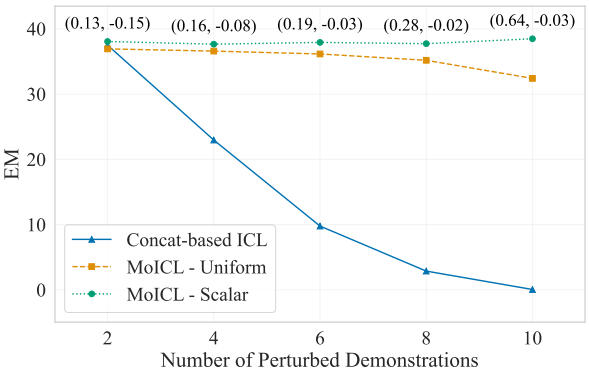


Figure 4: Resilience of ICL to adding perturbed demonstration. We report the EM based on the number of perturbed demonstrations out of the total 12 demonstrations in NQ. For the case of *scalar*, we also present the average weights of standard and perturbed demonstrations as (standard, perturbed).

3.7 Filtering Perturbed Demonstrations

One of the benefits of assigning weights to each demonstration or its subsets is the ability to handle low-quality, or more specifically, perturbed demonstrations. To verify this, in NQ-Open, we perturbed demonstrations (see Appendix B.4 for the result of NQ-open without perturbation) by randomly changing the answers to one of (*yes*, *no*, *foo*, *bar*), where the total number of demonstration is 12, and each subset has one demonstration (n, $k = 12$). The results in Fig. 4 show that our proposed method effectively handles perturbed demonstrations. While the EM of the concat-based ICL significantly decreases as the number of perturbed demonstrations increases, the MoICL methods can maintain performance. Additionally, without tuning the weights (*uniform*), performance gradually declines as the

Method ↓ Subset →	$k' = 5$	$k' = 10$	$k' = 20$	$k' = 30$	$k' = 90$
Concat-based ICL ($n = k'$)	72.19±2.63	74.12±2.24	74.84±1.88	76.44±2.48	75.67±2.33
MoICL ($n, k = k'$)					
uniform	73.05±0.52	73.42±0.76	73.42±0.49	73.37±0.34	73.26±0.16
scalar	**76.26**±**1.11**	**78.16**±**0.91**	**80.16**±**1.23**	**81.33**±**0.69**	**83.35**±**0.41**
MoICL *scalar* ($n = k = 90$)					
Highest k' Weights	75.58±0.81	75.56±0.46	74.42±0.61	74.33±0.38	-
Highest k' Weights (abs)	69.79±14.84	60.53±21.84	71.58±14.67	72.93±13.14	-
IMLE Top-k' mask	**76.07**±**0.64**	**75.93**±**0.69**	**76.35**±**0.35**	**76.44**±**0.64**	-

Table 6: Analysis of selecting useful demonstrations with the proposed MoICL on the TweetEval Offensive test set on Llama-3-8b-Instruct. 'Highest k' Weights' refers to selecting the k' subsets with the largest weights out of 90 weights of MoICL *scalar*, while 'Highest k' Weights (abs)' uses absolute weights instead.

Method ↓ Dataset →	**Offensive**	**Hate**	**SST**	**RTE**	**FEVER**	**PAWS**	**QNLI**
Concat-based ICL ($n = 30$)	76.44±2.48	53.54±4.29	**95.46**±**0.14**	**86.43**±**1.26**	80.63±0.49	**78.12**±**0.77**	89.08±0.44
Mixture of ICL ($n = k = 30$)							
uniform	73.37±0.34	59.12±0.47	94.17±0.21	77.26±1.02	79.46±0.36	65.29±0.51	88.66±0.25
adaptive	**77.33**±**3.28**	**65.01**±**4.10**	94.24±0.50	80.00±0.59	**80.68**±**0.32**	73.79±0.14	**89.21**±**0.26**

Table 7: Comparison of MoICL methods on unseen demonstrations, including *uniform* and *adaptive*, on the TweetEval Offensive and Hate, using Llama-3-8b-Instruct.

number of perturbed demonstrations increases, but with tuning (*scalar*), the performance remains stable (more than +35% with 10 perturbed demonstrations). This is clear when analysing the weights; Fig. 4 shows the average weights of normal and perturbed demonstrations, with significantly lower weights for the latter (e.g., 0.64 vs -0.03).

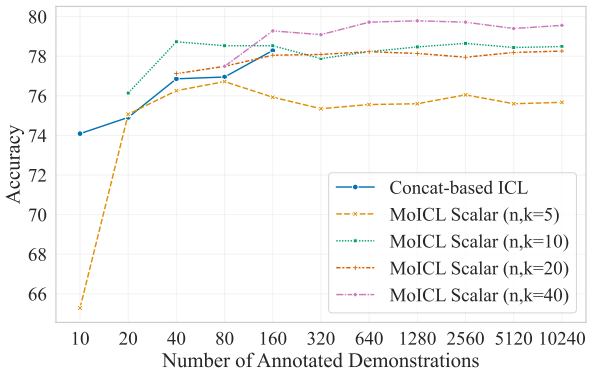
3.8 Selecting Demonstration Subsets

We now analyse the impact of sparsifying the mixture weights $w \in \mathbb{R}^k$ in MoICL. Results are available in Table 6 — "Highest n Weights" refers to selecting the subsets with the n largest w weights (or $|w|$ in the case of "abs"), while IMLE Top-k' mask refers to the method introduced in §2.2, following the default hyper-parameters proposed by Niepert et al. (2021). While MoICL (*scalar*) achieved the highest accuracy, the need to learn them for each m and k makes selection methods that tune weights for a large n and then select m of them more practical. Notably, "Highest n Weights (abs)" is high-variance, indicating the difficulty in effectively leveraging anti-experts (§3.3). In contrast, IMLE, which uses a mask, demonstrated stable performance, achieving the best results even with a limited number of demonstrations. This also suggests that MoICL can be interpreted through the lens of a mixture-of-experts paradigm: rather than relying solely on the highest-weighted experts, performance gains emerge from expert coordination and interaction among experts within a broader set.

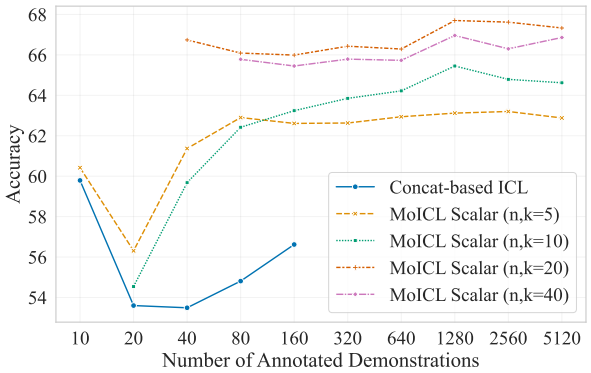
3.9 Generalisation to Unseen Demonstrations

While MoICL in the *scalar* setting is conceptually simpler, it has the disadvantage of requiring a fixed set of demonstration subsets. A solution to overcome this limitation is to utilise a smaller, fine-tuned hyper-network (*adaptive*) that calculates the weights for arbitrary demonstration subsets. Table 7 compares the performance of MoICL with different choices for the mixing weights in settings where the demonstration set D was not available during the training process. In this situation, *scalar*, which assumes that experts and their demonstrations are available beforehand, cannot be used. However, a hyper-network trained to map sets of demonstrations to mixture weights can generalise even when presented with unseen demonstrations.

Compared to the *uniform* baseline, MoICL consistently outperforms it, indicating that the hyper-network can assign meaningful, appropriate weights to arbitrary demonstration subsets. When compared to the concat-based ICL baseline, MoICL achieves competitive or superior performance on 4 out of 7 tasks. Considering that the concat-based ICL baseline uses full n-shot demonstrations while the adaptive variant of MoICL effectively operates in a much lower (n/k)-shot setting, this level of performance can be considered reasonable.

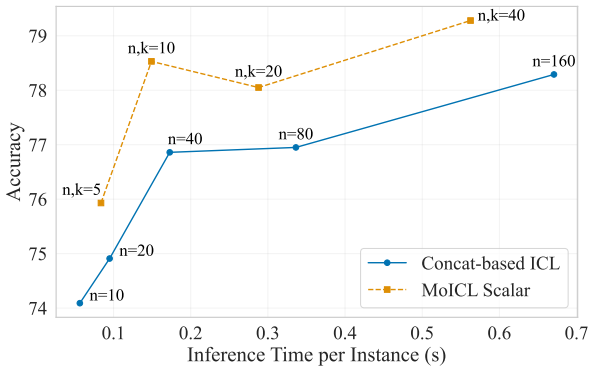


(a) TweetEval Offensive

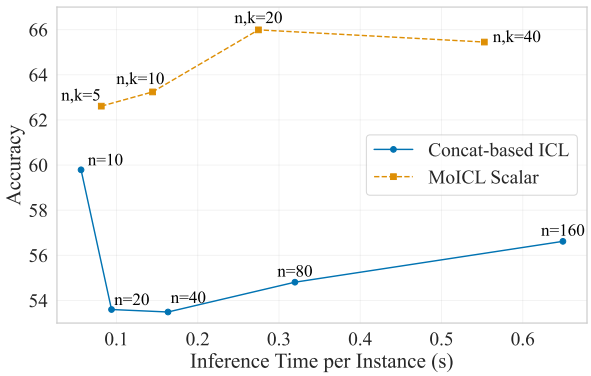


(b) TweetEval Hate

Figure 5: An analysis of MoICL's data-efficiency on the TweetEval offensive/hate test set using Llama-3-8B-Instruct. Concat-based ICL concatenated all available demonstrations (x-axis), exceeding the context length when ($n > 160$).



(a) TweetEval Offensive



(b) TweetEval Hate

Figure 6: An analysis of the inference time efficiency of MoICL on the TweetEval offensive/hate test set using Llama-3-8B-Instruct. The total number of demonstrations available to MoICL *scalar* is 160, the same as the maximum number of demonstrations that concat-based ICL can use within the context length limit.

4 Data and Compute Efficiency

To analyse the data-efficiency of MoICL, we present the accuracy on TweetEval Offensive and Hate test set in Fig. 5 under settings where the number of training instances (Number of Annotated Demonstrations) is limited. In this experiment, we set $n = k$, so each expert is assigned one demonstration and weight tuning is performed using the number of training instances minus k (e.g., when the x-axis is at 40, MoICL with $k = 10$ is tuned with 30 training instances). We observed that MoICL is highly data-efficient, achieving better performance than concat-based ICL with only around 20 annotated demonstrations. In contrast, concat-based ICL showed lower performance when given the same number of annotated demonstrations and particularly struggled when the number of demonstrations exceeded 160, exceeding the maximum allowed context length.

Furthermore, we analyse whether MoICL is more time-efficient than concat-based ICL. Fig. 6 compares the performance in terms of the average inference time (in seconds) per instance with up to 160 annotated demonstrations (which is the context length limit for concat-based ICL) are provided. We observed that MoICL consistently showed higher accuracy compared to concat-based ICL relative to inference time, demonstrating that MoICL is not only data-efficient but also time-efficient. In addition, Appendix B.5 shows that MoICL training can also be made more efficient with caching.

Complexity The proposed MoICL method partitions demonstrations into subsets rather than concatenating them, thereby reducing the input context length for LLMs. This reduction is beneficial in Transformer-based architectures, where computational load increases quadratically with the context length. In Table 8, we analyse the computation cost based on the unit computation cost (one forward pass for one example) of an LLM and a hyper-network, namely C_{LLM} and C_{Hyper}. Concat-based ICL exhibits the highest cost by concatenating all demonstrations and the test input, whereas Ensemble-based ICL shows the lowest cost by concatenating each demonstration with the test input. MoICL lies in-between, with the

Method	Complexity
Concat-based ICL	$(n+1)^2 \cdot C_{\text{LLM}}$
Ensemble-based ICL	$n \cdot (1+1)^2 \cdot C_{\text{LLM}}$
Mixture of ICL	
uniform	$k \cdot (\frac{n}{k}+1)^2 \cdot C_{\text{LLM}}$
scalar	$k \cdot (\frac{n}{k}+1)^2 \cdot C_{\text{LLM}}$
adaptive	$k \cdot (\frac{n}{k}+1)^2 \cdot C_{\text{LLM}} + n^2 \cdot C_{\text{Hyper}}$

Table 8: Comparison of the computational complexity at inference time between MoICL Methods and Baseline ICL Methods. C_{LLM} and C_{Hyper} refer to the unit computation complexity for one demonstration and one forward pass for an LLM and a hyper-network, respectively. n and k refer to the number of demonstrations and the number of subsets.

cost determined by the number of subsets k. In *adaptive*, the weights calculation adds a cost of $(n+1)^2 \cdot C_{\text{Hyper}}$. Since C_{LLM} is usually significantly larger than C_{Hyper}, this approach still offers a computational advantage. Furthermore, the weights of the experts need to be computed only once, which means $n^2 \cdot C_{\text{Hyper}}$ is a one-time cost.

5 Related Work

In-Context Learning In-context learning (ICL) is an approach to few-shot learning by concatenating the training examples and providing them as input to the model before the actual test example. Being able to perform ICL is an *emerging ability* of very large models, such as GPT-3 (Brown et al., 2020) and PaLM (Chowdhery et al., 2023). One characteristic of ICL is that increasing the number of demonstrations tends to increase the downstream task accuracy (Brown et al., 2020; Lu et al., 2022). However, Agarwal et al. (2024) show that, after a given number of demonstrations, performance saturates and additional examples might even decrease the downstream task accuracy. Furthermore, in Transformer-based LLMs, increasing the number of ICL demonstrations can be too computationally demanding due to the complexity of self-attention operations growing quadratically with the context size (Liu et al., 2022). Finally, ICL is sensitive to out-of-domain demonstrations (Min et al., 2022b) or label imbalance, underscoring the importance of the selection of the in-context demonstrations to use (Zhao et al., 2021; Fei et al., 2023).

Ensembles of Demonstrations Min et al. (2022a) introduce *ensemble-based demonstrations*, where each demonstration is provided to a language model along with the input to obtain a next-token distribution; such next-token distributions are then combined in a product-of-experts. Le et al. (2022) propose Mixtures of In-Context Experts for anaphora resolution, where the weights for each expert were calculated based on the cosine similarity between the embeddings of the test input and the demonstrations. Ye et al. (2023) extend the models by Le et al. (2022) and analyse the impact of merging the expert activations at different stages, both in terms of efficiency and downstream task performance. Wang et al. (2024) proposed a framework that partitions demonstrations into semantically similar clusters, assigns them to experts, and trains a retriever model for each expert to retrieve demonstrations based on a given input query.

MoICL is related to ensemble-based ICL methods in that it aggregates predictions from multiple demonstration subsets. However, unlike prior approaches that rely on similarity-based retrieval or uniform aggregation, MoICL learns performance-driven weights across fixed or dynamic subsets, enabling more explicit modelling of expert contributions. Most notably, Huang et al. (2025) (LARA) shares similar motivations and can be considered as a concurrent approach. While LARA uses non-gradient-based optimisation, MoICL leverages gradient-based optimisation with support for supervised fine-tuning and preference tuning (Appendix B.7). Additionally, MoICL incorporates a hyper-network (*adaptive*) to generalise to unseen demonstration subsets (Section 3.9), which addresses the limitation of requiring fixed demonstrations between training and inference. MoICL also introduces the concept of anti-experts in ICL and demonstrates practical effectiveness under challenging conditions such as out-of-domain inputs, label imbalance, and perturbed demonstrations.

6 Conclusions

We proposed MoICL, a method for dynamically learning to combine multiple models, each trained via ICL, via gradient-based optimisation methods. We show that MoICL significantly improves accuracy compared to several strong baselines, as well as in terms of data, memory, and compute efficiency. Furthermore, we show that MoICL is robust to out-of-domain and perturbed demonstrations, can help mitigate label imbalance, and can be used to select sets of demonstrations.

Limitations

Although MoICL does not require direct access to the model parameters, it requires access to the logits of the distribution over the vocabulary or answers produced by the model, both to train the experts and to calculate the final prediction at inference time, which prevents its use with completely black-box models like GPT-4. As discussed and tested in Appendix B.6, it is possible to apply MoICL by leveraging an approximation of the full logits in situations where logits are extracted from proprietary, restricted-access models or only top-k tokens are available. However, the requirement for logits themselves (even if only partially) can be a limitation.

An important direction for future work, though not explored in this study, is training scalar weights for all demonstrations in the training set. Currently, we sample n demonstrations from the training set and assign them to experts, tuning their weights. Extending this to all demonstrations in the training set would require progressively expanding the experts and their tuned weights. One possible approach for future work is to incorporate the search and relevance heuristics proposed by Li and Qiu (2023) as inductive biases in our proposed MoICL (*adaptive*).

Additionally, due to computational resource limitations, we conduct our experiments on the Llama-2 models (Llama-2-7B-chat, Llama-2-13B-chat, Llama-2-70B-chat), Llama-3 models (Llama-3-8B, Llama-3-8B-Instruct), Llama-3.1 models (Llama-3.1-8B, Llama-3.1-8B-Instruct), and Llama-3.2 models (Llama-3.2-3B, Llama-3.2-1B, Llama-3.2-1B-Instruct) as target LLMs, and T5-models (T5-efficient-tiny, T5-efficient-mini, T5-small, T5-base) as hyper-network models (Appendix B.1). However, our method is not limited to specific LMs and can be applied across various models.

Acknowledgments We thank the anonymous reviewers for their useful feedback and comments. Giwon Hong was supported by the ILCC PhD program (School of Informatics Funding Package) at the University of Edinburgh, School of Informatics. Pasquale Minervini and Emile van Krieken were partially funded by ELIAI (The Edinburgh Laboratory for Integrated Artificial Intelligence), EPSRC (grant no. EP/W002876/1). Additionally, Pasquale Minervini was partially funded by an industry grant from Cisco and a donation from Accenture LLP. This work was supported by the Edinburgh International Data Facility (EIDF) and the Data-Driven Innovation Programme at the University of Edinburgh.

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie CY Chan, Biao Zhang, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. In *ICML 2024 Workshop on In-Context Learning*.

AI@Meta. 2024. Llama 3 model card.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

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. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad

Nasr, Arthur Conmy, et al. Stealing part of a production language model. In *Forty-first International Conference on Machine Learning*.

Jiuhai Chen, Lichang Chen, Chen Zhu, and Tianyi Zhou. 2023a. How many demonstrations do you need for in-context learning? In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Yanda Chen, Chen Zhao, Zhou Yu, Kathleen McKeown, and He He. 2023b. On the relation between sensitivity and accuracy in in-context learning. In *2023 Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 155–167. Association for Computational Linguistics (ACL).

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. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Yupei Du and Dong Nguyen. 2023. Measuring the instability of fine-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6209–6230, Toronto, Canada. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yu Fei, Yifan Hou, Zeming Chen, and Antoine Bosselut. 2023. Mitigating label biases for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14014–14031, Toronto, Canada. Association for Computational Linguistics.

Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. 2024. Logits of API-protected LLMs leak proprietary information. In *First Conference on Language Modeling*.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *ICLR (Poster)*. OpenReview.net.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Chengsong Huang, Langlin Huang, and Jiaxin Huang. 2025. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Hao Peng, Ximing Lu, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022. Twist decoding: Diverse generators guide each other. In *EMNLP*, pages 4909–4923. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Nghia T. Le, Fan Bai, and Alan Ritter. 2022. Few-shot anaphora resolution in scientific protocols via mixtures of in-context experts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2693–2706, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Xiaonan Li and Xipeng Qiu. 2023. Finding support examples for in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6219–6235.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. In *First Conference on Language Modeling*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Yao Lu, Jiayi Wang, Raphael Tang, Sebastian Riedel, and Pontus Stenetorp. 2024. Strings from the library of babel: Random sampling as a strong baseline for prompt optimisation. In *NAACL-HLT*, pages 2221–2231. Association for Computational Linguistics.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pasquale Minervini, Luca Franceschi, and Mathias Niepert. 2023. Adaptive perturbation-based gradient estimation for discrete latent variable models. In *AAAI*, pages 9200–9208. AAAI Press.

John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. 2024. Language model inversion. In *The Twelfth International Conference on Learning Representations*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. 2021. Implicit mle: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. In *ACL (Findings)*, pages 6287–6310. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Song Wang, Zihan Chen, Chengshuai Shi, Cong Shen, and Jundong Li. 2024. Mixture of demonstrations for in-context learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Xin Xu, Yue Liu, Panupong Pasupat, Mehran Kazemi, et al. 2024. In-context learning with retrieved demonstrations for language models: A survey. *arXiv preprint arXiv:2401.11624*.

Qinyuan Ye, Iz Beltagy, Matthew Peters, Xiang Ren, and Hannaneh Hajishirzi. 2023. FiD-ICL: A fusion-in-decoder approach for efficient in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8158–8185, Toronto, Canada. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *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)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR.

Split	Offensive	Hate	SST2	RTE	FEVER	PAWS	QNLI
Train set	11,916	9,000	66,349	2,190	54,550	49,401	99,743
Dev set	1,324	1,000	1,000	300	5,000	8,000	5,000
Test set	860	2,970	872	277	13,332	8,000	5463

Table 9: Statistics of the classification datasets used in our experiments.

A Detailed Experiment Settings

A.1 Datasets

TweetEval (Barbieri et al., 2020) offensive/hate datasets are originally from Zampieri et al. (2019) and Basile et al. (2019), respectively. PAWS (Zhang et al., 2019) is released under a custom license[5] from Google LLC. For SST-2 (Socher et al., 2013)[6], RTE (Bentivogli et al., 2009), FEVER (Thorne et al., 2018)[7], and QNLI (Wang et al., 2018)[8], we used the original validation/development set as the test set and sampled a portion of the training set to construct a new validation set. Table 9 presents the dataset split statistics for all classification datasets used in our experiments. For NQ-open (Lee et al., 2019)[9], we used the top 1 retrieved documents as a context. The dataset contains 79,168 train instances, 8,757 validation instances, and 3,610 test instances. GSM8K (Cobbe et al., 2021)[10] contains 7,473 train instances and 1,319 test instances, employing chain-of-thought (CoT). HH-RLHF (Bai et al., 2022)[11] Harmless dataset contains 42,537 training instances (chosen, rejected pairs) and 2,312 test instances.

A.2 Hyperparameters

We used five seeds $[31, 42, 65, 438, 991]$ in all experiments except for Appendix B.7, which were applied to every possible aspect, including dataset shuffle, demonstration pooling and partition, and hyper-network fine-tuning, and baseline results. Also, we set the batch size to 1, the gradient accumulation steps to 12, the warmup steps to 500, the weight decay to 0.01, and the learning rate to 0.0001 without performing a hyperparameter search for these settings. We used AdamW (Loshchilov and Hutter, 2019) as an optimiser. We use the same setting for Direct Preference Optimization (DPO; Rafailov et al., 2024) in Appendix B.7 as well. For the PEFT (LoRA, Hu et al., 2022) baseline, we set r=16 (rank), alpha=32 (scale factor), and dropout=0.1. We did not perform a search for these LoRA hyperparameters as we utilised the default settings provided by Mangrulkar et al. (2022). Unless otherwise specified, a total of 30 demonstrations were used along with Static partitioning. Both baselines (when applicable) and MoICL are tuned for 5 epochs. We use greedy decoding for generating from MoICL. The hyper-networks used in our experiments have parameters of 16M (t5-efficient-tiny), 31M (t5-efficient-mini), 60M (t5-small), and 220M (t5-base), respectively.

A.3 Implementation Details

MoICL For all datasets used in the experiments, we fine-tuned all the MoICL weights and hyper-networks on the training set and evaluated them on the validation/development set at each epoch, selecting the ones with the highest performance. The results reported in all experiments were measured on the test set. For *scalar*, we first sampled D from the training set based on the different seeds and used the remaining training instances as D_T (§2.2). In experiments across multiple models (§3.4), the same fixed in-context demonstration set ($n = 6, k = 1$) was provided to all three models (Llama-3.2-1B, Llama-3.2-1B-Instruct, and Llama-3.2-3B), forming three separate experts. The outputs of these experts were then combined using a learned weighting function to generate the final output. For *adaptive* in §3.9, D is not available during training and is used only during evaluation. We further separate D_T into D_{pool} and D_{pair} randomly at each epoch, where demonstrations are sampled from D_{pool} and $(x, y) \in D_{pair}$. While any model that produces weights can be used for the hyper-network, we attach a linear layer on top of a pre-trained encoder-decoder T5-small (Raffel et al., 2020) model. For the data efficiency analysis on Fig. 5 in §4, we applied the same training step (10,240) to all different MoICL settings.

Baselines For PEFT fine-tuned on RTE, we applied early stopping based on the dev set accuracy, as we observed that the training process was highly unstable. Both Ensemble-based ICL (Min et al., 2022a) and LENS (Li and Qiu, 2023) used the Direct method instead of the Channel method,

[5]The dataset is provided "AS IS" without any warranty, express or implied. Google disclaims all liability for any damages, direct or indirect, resulting from the use of the dataset.
[6]The dataset is released under `The MIT License.`
[7]The dataset is released under `CC BY-SA 3.0` license.
[8]The dataset is released under `CC BY-SA 4.0` license.
[9]The dataset is released under `CC BY-SA 3.0` license.
[10]The dataset is released under `The MIT License.`
[11]The dataset is released under `The MIT License.`

Method ↓ Model →	ll2-chat-7b	ll2-chat-13b	ll2-chat-70b
Concat-based ICL	73.09±3.21	63.09±3.85	69.42±1.78
MoICL			
uniform	**79.35**±0.22	63.60±1.84	67.88±1.03
scalar	79.16±0.60	**80.49**±1.01	**82.26**±0.65

Table 10: Comparison on the TweetEval Offensive Test set across different sizes of the Llama-2 models.

Hyper-network Model	Offensive	Hate
t5-efficient-tiny (16M)	69.32±2.07 | 74.60±2.03	67.32±0.66 | 60.48±4.56
t5-efficient-mini (31M)	68.50±2.01 | 73.74±1.43	66.00±1.51 | 56.61±0.90
t5-small (60M)	**71.01**±1.09 | **76.65**±1.31	**70.20**±1.53 | **65.07**±5.22
t5-base (220M)	69.14±1.01 | 74.40±2.39	68.24±0.75 | 63.23±4.51

Table 11: Comparison on the TweetEval Offensive/hate Dev\|Test set using Llama-3-8b-Instruct as a target LLM across different sizes of the hyper-network. The numbers in parentheses indicate the number of parameters.

MoICL Method	Static	Random Size	BM25
uniform			
$k = 3$	**74.86**±1.84	**74.74**±1.90	**74.79**±1.79
$k = 5$	73.77±1.60	74.09±1.35	73.47±2.19
$k = 10$	74.00±0.87	73.37±0.94	74.40±0.82
scalar			
$k = 3$	76.14±1.48	77.37±1.97	77.21±2.02
$k = 5$	78.35±1.49	77.67±2.69	78.37±1.62
$k = 10$	**79.42**±1.48	**78.72**±0.87	**79.70**±1.32

Table 12: Analysis of partitioning methods on TweetEval Offensive dataset. Random and BM25 represent random clustering and clustering based on BM25 scores, respectively. Bold text signifies the highest accuracy for each method.

which also applied for MoICL as well. For LENS, We first apply *Progressive Example Filtering* to select 30 demonstrations, then perform *Diversity-Guided Search* to obtain five permutations of the examples and report the average and standard deviation based on these five permutations. For Le et al. (2022), we used SBERT model with *all-roberta-large-v1* checkpoint[12] to calculate input–demonstration similarity, following the original setting. For LARA (Huang et al., 2025), we used the original implementation provided by the authors[13], set n=k=30 (1-shot), increased the number of iterations from 20 to 40 (considering that switching from 2-shots to 1-shot increases the number of weights to learn), and modified the training instances to be a separate n×3 rather than n.

B Additional Analyses

B.1 Impact of Model Size

Considering the ongoing trend of scaling up LLMs, it is essential to analyse how the proposed method is affected by model size. In Table 10, we compare the accuracy of our proposed method on the TweetEval Offensive task when using Llama-2-chat models in various sizes (7B, 13B, 70B) as the target LLM. [14] Although the performance of the Llama-2-7B-chat model is somewhat unusual compared to the other two models, we observed that MoICL consistently outperforms concat-based ICL across all three model sizes.

We also analysed the impact of hyper-network model size on the performance of *adaptive* weights. Table 11 compares the dev/test set accuracy on the TweetEval hate/offensive task based on the size of the T5 model used as the hyper-network. We chose T5 because it is available in a wide range of model sizes, allowing us to systematically examine the tradeoff between model capacity and performance. Importantly, the hyper-network must remain significantly smaller than the underlying LLM to maintain overall efficiency. From analysing the dev set results, we found that even with a very small model size (16M–60M), the hyper-network performed reasonably well. Based on this analysis, we selected T5-small (60M) as the default hyper-network, as it offers a favourable balance between computational efficiency and performance, representing less than 1% of the total size of the main model (8B).

B.2 Partitioning a Demonstration set D.

In this work, we analyse the following partitioning strategies: **Static**, **Random Size**, and **BM25**. **Static** means partitioning n demonstrations into k subsets, with each subset containing n/k demonstrations. The demonstrations are selected randomly, but the subset sizes are kept fixed (hence, "static"). **Random Size** refers to partitioning into k subsets, but both the subset sizes and the demonstration assignments are randomised. **BM25** apply k-NN clustering based on BM25 scores over the demonstrations (Robertson et al., 2009), producing k semantically coherent subsets with variable sizes.

Table 12 compares the performance of MoICL methods and different partitioning methods (Static, Random, BM25) for the same k (number of subsets). In *uniform*, there is little difference between

[12] https://www.sbert.net/docs/sentence_transformer/pretrained_models.html

[13] https://github.com/Chengsong-Huang/LARA

[14] We chose LLaMA-2-chat because it offers three clearly separated model sizes, making it suitable for analysing trends across scale. In contrast, LLaMA-3 currently provides only 8B and 70B models (while LLaMA-3.1 includes a 405B version, it was beyond our available computational resources)

MoICL ↓ Label Ratio →	27:3	25:5	23:7
scalar (n=30, k=10)			
- Static Partitioning	73.33±0.96	72.72±0.76	73.81±2.25
- BM25 Partitioning	74.63±1.03	74.09±1.07	74.42±1.50

Table 13: Performance comparison of static and BM25-based partitioning under varying levels of label imbalance on the TweetEval Offensive dataset with $k = 10$ subsets. Each imbalance setting denotes the number of demonstrations per label (e.g., 27:3 indicates 27 for the majority label and 3 for the minority).

Static and Random and only a slight performance improvement with BM25. However, there is a common performance enhancement when MoICL *scalar* are applied. This indicates that our proposed method is not significantly affected by partitioning methods and can be applied in a complementary manner across them. As such, we decided to use only the Static method in the other experiments.

To further assess the robustness of partitioning strategies, we investigate their behaviour under label imbalance (§3.6). Specifically, we compare static and BM25-based partitioning on the TweetEval Offensive dataset using $k = 10$ subsets, with varying degrees of label imbalance across demonstrations (from 27:3 to 23:7 ratios). As shown in Table 13, BM25 consistently outperforms the static strategy, especially in highly imbalanced settings such as 27:3 and 25:5. We attribute this to BM25's ability to group demonstrations based on semantic similarity, which often correlates with label similarity. This clustering effect enables MoICL to assign more targeted and informative weights to subsets, thereby mitigating the negative impact of label imbalance. These findings suggest that while static partitioning is effective and stable in general, semantically informed strategies like BM25 can offer clear advantages when label distributions are skewed.

B.3 Logits vs. Probabilities for Mixing Experts

As stated in §2.2, we mix the experts in the log domain. However, it is also possible—and perhaps more appropriate—to use a regular mixture of probabilities, as in Eq. (3).

$$p\left(y \mid D, x\right) \propto \left[\sum_{i=1}^{k} w_i p\left(y \mid D_i, x\right)\right] \quad (3)$$

Accordingly, in Fig. 2, we compare the accuracy trends based on partitioning size when using weighting in the probability and logit domains.

Methods ($n = 12$)	GSM8K (Acc.)
Concat-based ICL	61.83±1.77
Mixture of ICL (*uniform*)	
$k = 6$	62.32±1.25
$k = 12$	61.74±0.78
Mixture of ICL (*scalar*)	
$k = 6$	63.20±0.50
$k = 12$	62.14±1.03
Mixture of ICL (*adaptive*)	
$k = 6$	**63.96**±2.00
$k = 12$	61.49±2.19

Table 14: Comparison between baseline methods and MoICL on GSM8K using Llama-3-8B. k represents the number of demonstrations subset, where the total number of demonstrations (n) is 12

In *uniform*, whether logits or probabilities were used did not make a significant difference, but in *scalar*, the impact was substantial. This is likely because distinct differences in the distribution patterns among experts (and thus useful information in the mixture) get diluted during the normalisation process when using probabilities.

B.4 MoICL in a Generation Task

In addition to the classification tasks in §3.1, we also apply our MoICL on a generation task, NQ-open (Lee et al., 2019) and GSM8K (Cobbe et al., 2021), in Table 14 and Table 15. However, unlike in classification tasks, MoICL did not show significant EM improvements over baseline approaches in NQ. For GSM8K, although MoICL using *scalar* weights and *adaptive* weights achieves performance improvements over Concat-based ICL and uniform weighting, these gains are not statistically significant when considering the standard deviations. Nevertheless, as seen in §3.7, MoICL exhibited strong robustness in situations involving perturbed demonstrations, proving the usefulness of the expert's tuned weights.

B.5 Reducing Training Costs Through Caching

While MoICL does not fine-tune the expert models themselves, it leverages their next-token distributions to guide the weighting function. A key efficiency gain arises from caching these distributions during the training process. Because the underlying LLM parameters remain fixed, the output distributions computed at the outset can be stored and subsequently reused for all training instances.

Methods	NQ-open (EM)
Concat-based ICL	
$n = 12$	40.34±0.26
$n = 24$	40.58±0.47
$n = 48$	40.07±0.50
$n = 96$	-
Concat-based ICL (Llama-3.1-8B)	
$n = 12$	39.70±0.56
$n = 24$	40.29±0.75
$n = 48$	40.17±0.58
$n = 96$	40.47±0.68
Mixture of ICL (*scalar*, $n = 96$)	
$k = 6$	**40.86±0.31**
$k = 12$	40.74±0.35
Mixture of ICL (*adaptive*, $n = 96$)	
$k = 6$	40.66±0.26
$k = 12$	40.64±0.48

Table 15: Comparison between baseline methods and MOICL on NQ-open using Llama-3-8B. We also use Llama-3.1-8B for concat-based ICL to evaluate the effectiveness of a longer-context model. k represents the number of demonstrations subset, where the total number of demonstrations is n. For $n = 96$, the performance of concat-based ICL with Llama-3-8B could not be measured due to exceeding the context length limit.

This approach eliminates the need to recompute the distributions from scratch for each training epoch.

Table 16 reports the number of floating-point operations (FLOPs) for MOICL 's training and inference on the offensive classification task, using a *scalar* weighting function, k=10, and 10 demonstrations for weight tuning[15]. Without caching, the first training epoch requires 21.61 TFLOPs per instance and 216.15 TFLOPs per epoch. By contrast, once caching is enabled, training costs drop drastically to approximately 205 FLOPs per instance and 2,050 FLOPs per epoch. Moreover, these training costs represent only about 1% of the total FLOPs when inference—requiring roughly 18,975.90 TFLOPs—is considered. By quantifying training costs in terms of FLOPs, we clearly demonstrate that freezing the LLM and leveraging caching yields substantial efficiency gains, making MOICL highly practical.

B.6 Applying MOICL to Black-Box Models

Proprietary LLMs offered via API often differ in the level of access they provide to their underlying distributions. While some grant full distributional access, others restrict output to the top-k logits, enable user-defined logit biases, or limit the user to text-only outputs without any direct probability information. Such constraints pose challenges for methods like MOICL, which rely on next-token distributions to tune the weighting function.

However, recent work has shown that extracting logits or other internal model components from proprietary, restricted-access models is feasible under a range of conditions. Morris et al. (2024) detail approaches to language model inversion, while Carlini et al. and Finlayson et al. (2024) explore methods for partially revealing models' logits and other parameters despite limited access. These techniques demonstrate that even under scenarios with partial distributional access, user-specified logit biases, or text-only responses, there are practical strategies to approximate or recover critical information about the model's outputs.

To investigate the applicability of MOICL in settings where only top-k logits are available, in Table 17, we trained and evaluated MOICL on the TweetEval Offensive and GSM8K tasks under a setting where only the top 40 logits were provided. MOICL demonstrates its performance under a black-box setting—where only partial distributional information is accessible—demonstrating its potential adaptability to restricted-access environments.

B.7 MOICL with Preference Fine-Tuning

One notable feature of MOICL is its ability to incorporate anti-experts into ICL, as discussed in §3.3. This opens up the possibility of applying MOICL to preference fine-tuning (e.g. for RLHF) where, for a given input query, there are preferred responses ("chosen") and dispreferred responses ("rejected"). MOICL can learn the weighting function of experts and anti-experts through Direct Preference Optimization (DPO; Rafailov et al., 2024) by assigning "chosen" demonstrations to experts and "rejected" demonstrations to anti-experts.

Table 18 compares the results of training and evaluating MOICL DPO on the HH-RLHF harmless dataset (Bai et al., 2022) with concat-based ICL and *uniform* weighting. When "chosen" and "rejected" demonstrations were assigned to experts and anti-experts, respectively (chosen + rejected), MOICL outperforms concat-based ICL by 2.1%. The learned weights of MOICL showed a clear distinction, with experts having weights of 0.43 and 0.60, while anti-experts had weights of 0.15

[15] In this setup, we use k=10 and 10 demonstrations (20 annotated demonstrations in total) for weight tuning, as these values have been found (in §4 and Fig. 5) to provide a practical balance between performance and computational cost.

	Epoch w/o cache (1st epoch)	Epoch w/ cache (2nd-last epoch)
Training FLOPs per Instance	21.61 TFLOPs	205 FLOPs
Training FLOPs per Epoch	216.15 TFLOPs	2,050 FLOPs
Inference FLOPs per Instance	22.07 TFLOPs	-
Inference FLOPs in Total	18,975.90 TFLOPs	-
FLOPs in total (Training + Inference)	19,191.91 TFLOPs	-

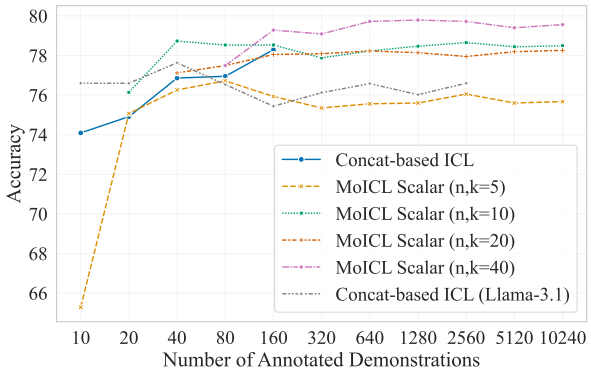
Table 16: Training and inference FLOPs for MoICL on the offensive classification task. We report per-instance and per-epoch costs both with and without caching, using a *scalar* weighting function, k=10, and 10 demonstrations for weight tuning.

	Offensive	GSM8K
MoICL (*scalar*, $n = k$)	81.33$\pm$0.69	62.14$\pm$1.03
w/ black-box setting	79.81$\pm$0.39	62.26$\pm$1.12

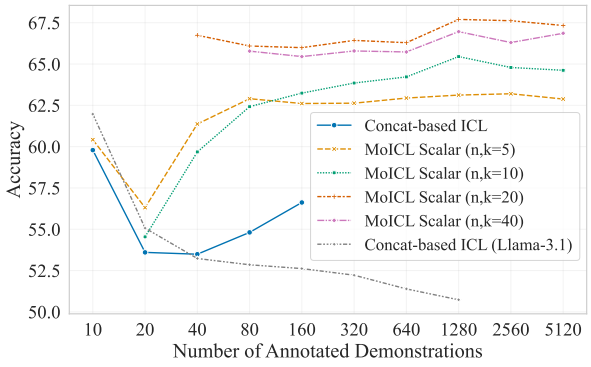
Table 17: MoICL on the TweetEval Offensive and GSM8K tasks using Llama-3-8B-Instruct under a black-box scenario where only the top-40 logits are accessible.

	HH-RLHF Harmless
Concat-based ICL (n=8)	55.02
MoICL ($n = 2, k = 4$)	
uniform	55.41
adaptive (chosen only)	56.23
adaptive (chosen + rejected)	**57.14**

Table 18: Results (accuracy) of MoICL using Direct Preference Optimization (DPO) on the HH-RLHF Harmless dataset test set using Llama-3-8B-Instruct. "chosen + rejected" represents chosen and rejected demonstrations assigned to experts ($k = 2$) and anti-experts ($k = 2$), respectively.



(a) TweetEval Offensive



(b) TweetEval Hate

Figure 7: An analysis of MoICL's data-efficiency on the TweetEval offensive/hate test set using Llama-3-8B-Instruct. Concat-based ICL concatenated all available demonstrations (x-axis), exceeding the context length when ($n > 160$). We also use Llama-3.1-8B-Instruct for concat-based ICL to evaluate the effectiveness of a longer-context model.

and -0.32. This result demonstrates that MoICL can successfully incorporate anti-experts into ICL, proving its potential for extension to preference tuning.

B.8 Comparison with Longer Context Models

By distributing demonstrations across experts or using them for tuning the weight function, MoICL can mitigate the limitations imposed by the model's context length. However, this limitation could also be addressed by using an LLM with a longer context length. To evaluate whether MoICL remains effective when compared to longer-context LLMs, we assess the performance of longer-context LLMs (Llama-3.1-8B and Llama-3.1-8B-Instruct with a 128k context length) to MoICL based on the original base LLMs (Llama-3-8B and Llama-3-8B-Instruct with an 8k context length) in scenarios where the context length of the base LLMs was exceeded.

As shown in Table 15, on NQ-open, Llama-3-8B model fails to produce an output when $n = 96$ due to context length limitations. However, MoICL efficiently distributes demonstrations across experts ($k = 6$, $k = 12$), achieving effective performance that even outperforms the longer-context model, Llama-3.1-8B. Similarly, as shown in Fig. 7 for the TweetEval Offensive and Hate datasets, Llama-3-8B-Instruct model fails to produce an output when $n > 160$ due to context length limitations. While the longer-context model, Llama-3.1-8B-Instruct, maintained reasonable performance even at higher

n values (except when results could not be obtained due to memory limitations at $n > 2,560$ for Offensive and $n > 1,280$ for Hate), MoICL outperforms the longer-context model on both datasets.

One possible explanation for this is that simply increasing the number of demonstrations in ICL does not always lead to better performance, as performance may saturate or even degrade (Chen et al., 2023a). In this regard, MoICL 's effectiveness compared to longer-context models may be attributed to its more efficient ICL framework, which distributes demonstrations across experts and utilises them during training.

C Prompt Templates

Table 19 presents the corresponding metric and prompt template for all tasks included in the experiments. For NQ, the delimiter for ICL demonstrations was '\n\n'. For the remaining tasks, '\n' was used as the delimiter.

D Computation Details

The experiments were conducted using NVIDIA A100 40GBs and 80GBs with 120GB of RAM. The GPU hours vary depending on the models and tasks; tuning MoICL scalar weights ($n, k = 30$) on TweetEval offensive takes approximately 1 hour and 20 minutes per epoch.

Table 19: Prompt template setting details for the tasks. The double curly braces "{{}}" signify input data.

Task	Metric	Prompt Template
TweetEval Offensive	Accuracy	Classify tweets that are offensive as offensive, and tweets that are not offensive as neutral. {{*ICL Demonstrations*}} Tweet: {{*tweet*}} Label:
TweetEval Hate	Accuracy	Classify tweets that are hateful against immigrants or women as hate and tweets that are not hateful against immigrants or women as neutral. {{*ICL Demonstrations*}} Tweet: {{*tweet*}} Label:
SST2	Accuracy	Classify sentences that are negative as negative and sentences that are positive as positive. {{*ICL Demonstrations*}} Sentence: {{*sentence*}} Label:
RTE	Accuracy	Classify two sentences that entail each other as true and two sentences that do not entail each other as false. {{*ICL Demonstrations*}} Sentence1: {{*first sentence*}} Sentence2: {{*second sentence*}} Label:
FEVER	Accuracy	Classify claims that are false as refuted, and tweets that are true as supported. {{*ICL Demonstrations*}} Claim: {{*claim*}} Label:
PAWS	Accuracy	Classify the two sentences as yes if they are paraphrases of each other, and if not, classify them as no. {{*ICL Demonstrations*}} sentence1: {{*first sentence*}} sentence2: {{*second sentence*}} label:
QNLI	Accuracy	Classify as yes if the sentence contains the answer to the question, if not, classify as no. {{*ICL Demonstrations*}} sentence: {{*sentence*}} question: {{*question*}} label:
NQ	EM	{{ICL Demonstrations}} title: {{*title*}} text: {{*text*}} Question: {{*question*}} Answer:
GSM8K	EM	{{ICL Demonstrations}} Q: {{*question*}} A: {{CoT *reasoning*}} The answer is
HH-RLHF	Accuracy	{{ICL Demonstrations}} {{*Chosen response*}} or {{*Rejected response*}}