# ADAPTIVE DEMONSTRATION ORDERING FOR IN-CONTEXT LEARNING VIA PERFORMANCE-ORIENTED ORDERING PREDICTOR

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In-context learning (ICL) is highly sensitive to demonstration ordering, with different orderings often leading to substantial variations in downstream performance. However, selecting an effective demonstration ordering is challenging because its downstream utility depends jointly on the input query, the retrieved demonstrations, and the target LLM. Therefore, we formulate demonstration ordering as a performance prediction task and propose Adaptive Demonstration Ordering for In-Context Learning (ADOICL). ADOICL is a performance-oriented and adaptive framework that selects demonstration orderings conditioned on the input query, the retrieved demonstrations, and the target LLM. Specifically, ADOICL learns an ordering predictor from offline performance supervision to estimate the downstream performance of candidate demonstration orderings and select the most promising demonstration ordering before a single target-LLM inference. We instantiate ADOICL as ADOICLS, a sequence-level scorer for serialized query–ordering pairs, and ADOICLE, an embedding-first scorer that reuses query and demonstration embeddings across candidate orderings to reduce encoding overhead and encoding time. Empirically, ADOICL achieves state-of-the-art performance across 9 target LLMs, 5 tasks, and 8 datasets, covering classification, generation, and regression tasks. Additional evaluations demonstrate ADOICL's plug-and-play compatibility with different retrieval pipelines and show that its predictor transfers across LLMs, datasets, and tasks. ADOICL also supports visual ICL, few-shot predictor training with only 30 queries, and many-shot demonstration settings with 100 demonstrations.

## 1 INTRODUCTION

In-context learning (ICL) is now a prevalent paradigm for adapting large language models (LLMs) to downstream tasks purely through in-context (the demonstrations retrieved from the retrieval dataset), without any parameter updates (Moreno-Ortiz & García-Gámez, 2026; Shi et al., 2026; Ott, 2026; Wang et al., 2026). However, ICL is highly sensitive to demonstration ordering: even with the same input query, retrieved demonstrations, and target LLM, different demonstration orderings can lead to substantially different inference outputs and task performance (Lu et al., 2022; Li et al., 2025b). Importantly, such ordering sensitivity is not merely random noise, but a stable and reproducible phenomenon observed across tasks, target LLMs, and datasets (Bhope et al., 2025). Therefore, effective demonstration ordering is crucial for improving ICL performance.

Existing demonstration-ordering methods use criteria such as query–demonstration similarity, entropy, and output confidence to guide order selection (Liu et al., 2022; Lu et al., 2022; Liu et al., 2024; Bhope et al., 2025). Some methods adapt the ordering to individual queries (Guo et al., 2024; Bhope et al., 2025), but adaptivity alone does not ensure alignment with downstream task performance: an ordering favored by a proxy criterion may still produce an inferior task outcome. Moreover, methods that score demonstrations independently capture primarily individual or query–demonstration properties, without explicitly modeling interactions among demonstrations in the ordered context. Methods that evaluate complete prompts can reflect these interactions through target-model outputs, yet confidence or entropy remains an indirect measure of task quality. These distinctions matter because the effectiveness of an ordering depends jointly on the input query, the retrieved



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

demonstration set, and the target LLM. This motivates learning a query-conditioned, order-sensitive predictor from observed target-LLM task outcomes to guide candidate-order selection.

As shown in Figure 1, representative proxy-based methods (e.g., ICCL (Liu et al., 2024) and GlobalE (Lu et al., 2022)) do not consistently achieve strong empirical performance across target LLMs and datasets. Their performance remains close to random ordering and substantially below that of the best ordering identified through exhaustive search. These observations highlight two key findings: (i) a single static demonstration ordering strategy cannot perform well uniformly across different queries, retrieved demonstrations, and target LLMs; and (ii) commonly used proxy scores are poorly calibrated to the target LLM's true task performance, such that optimizing proxy objectives does not reliably translate into improved downstream outcomes.

More generally, applying a fixed ordering strategy uniformly across different input queries is often suboptimal when strategies are evaluated by downstream performance. By analogy, prescribing the same medication to all patients is inappropriate; effective treatment must be adaptively tailored to each individual's condition and be clinically performance-oriented. Similarly, the empirical evidence in Figure 1 motivates the need for a more principled approach to demonstration ordering: **(i) adaptivity**, such that the ordering dynamically depends on the input query, the retrieved demonstrations, and the target LLM; and **(ii) performance orientation**, such that ordering decisions are guided directly by downstream task performance rather than task-agnostic proxy scores.

Motivated by the above observations, we propose **Adaptive Demonstration Ordering for In-Context Learning (ADOICL)**, a performance-oriented framework that formulates demonstration ordering as a performance prediction problem. Rather than relying on task-agnostic proxy scores, ADOICL directly predicts the downstream performance of candidate orderings conditioned on the input query, retrieved demonstrations, and target LLM, and selects the ordering with the highest predicted performance. Specifically, ADOICL trains a lightweight ordering predictor using offline performance supervision collected from the target LLM. At inference time, the predictor evaluates candidate demonstration orderings and selects the most promising one, after which the target LLM is queried *only once* using the selected ordering to produce the final prediction.

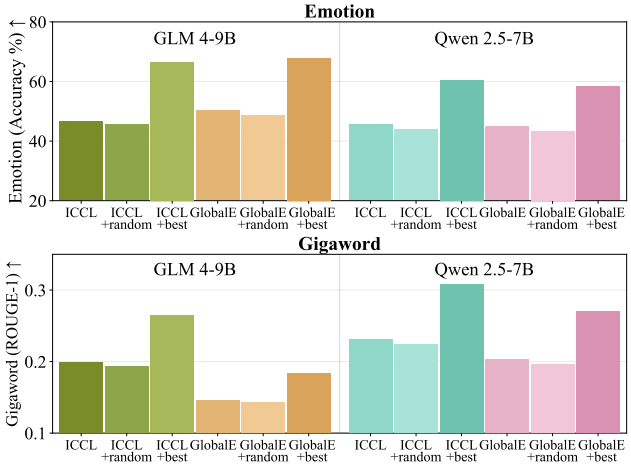


Figure 1: We evaluate GLM-4-9B and Qwen2.5-7B on Emotion (classification) and Gigaword (summarization) with $k$=4 demonstrations, comparing ICCL, GlobalE, their random-order variants (ICCL-random/GlobalE-random), and the optimal ordering obtained via exhaustive search (4! permutations; ICCL-best/GlobalE-best). Random variants permute demonstrations randomly, while best variants report the highest-performing order. Results show ICCL and GlobalE offer little improvement over random ordering and are far below the optimal ordering across evaluated models and datasets.

We instantiate ADOICL in two complementary forms. **ADOICLS** is a sequence-level version that directly scores serialized query–ordering pairs, providing a flexible implementation of the general performance-prediction framework. In contrast, **ADOICLE** adopts an embedding-first architecture that embeds each query and retrieved demonstration once and reuses their representations across candidate orderings.

A central challenge in demonstration ordering is its combinatorial search space: for a retrieved set of $k$ demonstrations, there are $k!$ possible demonstration orderings. ADOICL addresses this by leveraging supervised performance scores from a labeled retrieval dataset. During training, for each query, we sample a small subset of candidate orderings and obtain their realized task performance offline using the target LLM, providing direct supervision for learning the ordering predictor without exhaustively evaluating all $k!$ orderings for every training query. At inference time, ADOICL

scores a finite candidate pool, which may contain all permutations when $k$ is small or be constructed through sampling or group-level ordering when $k$ is large.

To further improve efficiency, ADOICLE adopts an embedding-first formulation. The query and each retrieved demonstration are embedded once, and each candidate-order representation is constructed by concatenating the cached demonstration embeddings according to the ordering. This reduces repeated demonstration-side embedding computations from $k!$ to $k$. For many-shot demonstration settings, we further introduce a grouping-based variant that clusters the retrieved demonstrations in the embedding space into $G$ semantically coherent groups and performs ordering at the group level using pooled embeddings, reducing the effective ordering size from $k$ to $G$. Within each group, demonstrations are arranged according to query–demonstration similarity. This clustering-based group-level variant serves as a practical scalability mechanism for many-shot settings rather than an exhaustive optimization over all $k!$ instance-level permutations.

Extensive experiments demonstrate that ADOICL is effective and broadly applicable across diverse settings. (1) It achieves strong performance on 9 target LLMs, 5 tasks, and 8 datasets, covering LLMs from 1B to 70B (GLM, LLaMA, Qwen, DeepSeek, and GPT-4o) and tasks including classification, generation, and regression. (2) It is **plug-and-play**, consistently improving existing ICL pipelines. (3) It exhibits strong **generalization** across target LLMs, datasets, tasks, and different demonstration numbers—for example, a predictor trained on classification transfers to summarization, and one trained on open-source LLMs transfers to GPT-4o. (4) ADOICL supports visual ICL and **few-shot training-data** settings and provides a clustering-based group-level scalability variant for **many-shot** demonstration settings.

Our contributions are summarized as follows: ❶ We show that demonstration ordering should be treated as an *adaptive and performance-oriented inference-time decision* in ICL, explaining why static, proxy-based ordering criteria often fail to generalize across inputs, tasks, and target LLMs. ❷ We propose Adaptive Demonstration Ordering for In-Context Learning (ADOICL), a performance-oriented demonstration-ordering framework that predicts the downstream performance of candidate orderings. We instantiate it as ADOICLS, a sequence-level scorer for serialized query–ordering pairs, and ADOICLE, an embedding-first scorer that reduces repeated demonstration-side embedding computations. ❸ We conduct extensive experiments across 9 target LLMs, 5 tasks, and 8 datasets, demonstrating strong performance and generalization across LLM families, datasets, tasks, and demonstration numbers, including transfer to the commercial GPT-4o model. We further demonstrate ADOICL's plug-and-play compatibility with existing ICL pipelines and its applicability to visual ICL, few-shot predictor training, and many-shot demonstration settings.

*Due to space constraints, we defer the discussion of related work to Appendix Section A.*

## 2 METHOD

ADOICL formulates demonstration ordering as a performance-prediction problem conditioned on the input query, the retrieved demonstrations, and the target LLM. Given an input query, retrieved demonstrations, and a target LLM, ADOICL estimates the downstream performance of candidate demonstration orderings and selects the candidate ordering with the best predicted performance within a finite candidate pool before one final target-LLM inference. Figure 2 summarizes the process of ADOICLE. ADOICLE consists of ❶ training dataset construction; ❷ candidate demonstration-order sampling and performance supervision; ❸ ordering predictor training; and ❹ test-time demonstration-order selection and final prediction.

### 2.1 TRAINING DATASET CONSTRUCTION

We construct a training dataset for learning the ordering predictor. Each instance contains an `input query`, `in-context` containing a candidate demonstration ordering, and a corresponding `performance score`. This section describes how to construct `input query–candidate ordering in-context–performance score` tuples. (1) **Query–In-Context Set Split.** ADOICL starts from a labeled retrieval dataset $\boldsymbol{D}_{\text{re}} = \{(x^i_{\text{re}}, y^i_{\text{re}})\}^w_{i=1}$ that is disjoint from the test dataset. Here, $x^i_{\text{re}}$ denotes the input text and $y^i_{\text{re}}$ its ground-truth label. We randomly partition $\boldsymbol{D}_{\text{re}}$ into a query set $\boldsymbol{D}_{\text{qu}} = \{(x^i_{\text{qu}}, y^i_{\text{qu}})\}^m_{i=1}$ and an in-context set $\boldsymbol{D}_{\text{in}} = \{(x^i_{\text{in}}, y^i_{\text{in}})\}^n_{i=1}$, containing $10\%$

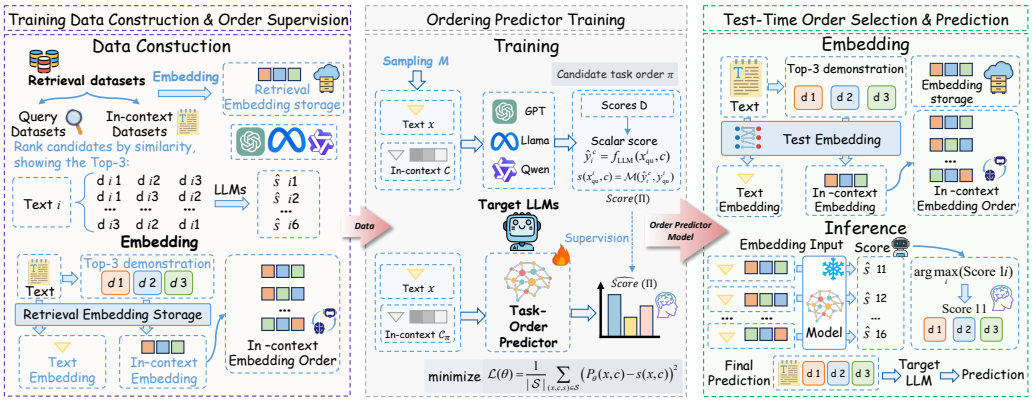


Figure 2: Overview of ADOICLE Instantiation for ADOICL. At *offline training* time, we retrieve a fixed demonstration set for each query, sample a small set of candidate demonstration orderings, and obtain performance scores by evaluating them with the target LLM. Then we train a lightweight ordering predictor under an *embedding-first* embedding that reuses demonstration embeddings. At *test time*, the ordering predictor selects the best ordering and queries the LLM once to produce the final output. ADOICLS replaces the embedding-based scorer with a sequence-level scorer.

and 90% of the instances, respectively. Formally,

$$\boldsymbol{D}_{\text{qu}} \cap \boldsymbol{D}_{\text{in}} = \varnothing,\ \boldsymbol{D}_{\text{qu}} \cup \boldsymbol{D}_{\text{in}} = \boldsymbol{D}_{\text{re}},\ m = 0.1w,\ n = 0.9w. \tag{1}$$

(2) **Demonstration Retrieval.** Given a text $x^i_{\text{qu}} \in \boldsymbol{D}_{\text{qu}}$, ADOICL retrieves a fixed set of candidate demonstrations from $\boldsymbol{D}_{\text{in}}$ using semantic similarity. Following standard practice Zhou et al. (2024); Liu et al. (2022), we use a pre-trained encoder $f_{\text{e}}(\cdot)$ to map each query text $x^i_{\text{qu}}$ and each candidate input $x^j_{\text{in}} \in \boldsymbol{D}_{\text{in}}$ to vector representations, *i.e.*, $\boldsymbol{v}^i_{\text{qu}} = f_{\text{e}}(x^i_{\text{qu}})$ and $\boldsymbol{v}^j_{\text{in}} = f_{\text{e}}(x^j_{\text{in}})$. Then, we compute the cosine similarity $\text{sim}\left(x^i_{\text{qu}}, x^j_{\text{in}}\right) = \frac{\boldsymbol{v}^i_{\text{qu}} \cdot \boldsymbol{v}^j_{\text{in}}}{\|\boldsymbol{v}^i_{\text{qu}}\| \|\boldsymbol{v}^j_{\text{in}}\|}$. We rank all candidates by cosine similarity and retrieve the top-$k$ demonstrations as $\mathcal{N}_k(x^i_{\text{qu}}) = \arg\max^k_{(x^j_{\text{in}}, y^j_{\text{in}}) \in \boldsymbol{D}_{\text{in}}} \text{sim}(x^i_{\text{qu}}, x^j_{\text{in}})$. For notational convenience, we re-index the retrieved demonstrations as

$$\mathcal{N}_k(x^i_{\text{qu}}) = \{(x^{i1}_{\text{in}}, y^{i1}_{\text{in}}), \ldots, (x^{ik}_{\text{in}}, y^{ik}_{\text{in}})\}. \tag{2}$$

Given the retrieved demonstration set $\mathcal{N}_k(x^i_{\text{qu}})$, different permutations correspond to distinct candidate demonstration orderings under a fixed query and target LLM. $\pi$ denotes a permutation operator that reorders the $k$ retrieved demonstrations. We denote the space of all possible orderings as

$$\mathcal{C}(x^i_{\text{qu}}) = \left\{\pi\left(\mathcal{N}_k(x^i_{\text{qu}})\right) \mid \pi \in \mathcal{P}_k\right\}, \tag{3}$$

where $\mathcal{P}_k$ denotes the permutation set over $k$ demonstrations and $|\mathcal{C}(x^i_{\text{qu}})| = k!$.

### 2.2 Candidate Demonstration Ordering Sampling and Performance Supervision

A core challenge in learning to rank demonstration orderings is obtaining informative supervision without evaluating all $k!$ possible permutations. As the ordering space $\mathcal{C}(x^i_{\text{qu}})$ grows factorially with the number of demonstrations $k$, exhaustively evaluating every candidate ordering with the target LLM is costly and becomes computationally infeasible as $k$ grows. We therefore adopt a sampling-based training strategy that provides tractable performance supervision from a small subset of candidate demonstration orderings. (1) **Candidate Demonstration Ordering Sampling.** For each training query $x^i_{\text{qu}}$, we sample a subset of candidate demonstration orderings:

$$\widetilde{\mathcal{C}}(x^i_{\text{qu}}) \subset \mathcal{C}(x^i_{\text{qu}}),\ \left|\widetilde{\mathcal{C}}(x^i_{\text{qu}})\right| = M,\ M \ll k!. \tag{4}$$

Candidate demonstration orderings are sampled as $M$ distinct elements of $\mathcal{C}(x^i_{\text{qu}})$, reducing the training cost from evaluating $k!$ orderings per query to only $M$ candidates. (2) **Performance-Based**

**Supervision.** For each sampled demonstration ordering $c \in \widetilde{\mathcal{C}}(x^i_{\text{qu}})$, we construct the corresponding in-context prompt and query the target LLM to obtain a prediction $\hat{y}^c_{i,\ell}$ and its target LLM and task-specific performance score $s_\ell(x^i_{\text{qu}}, c)$:

$$\hat{y}^c_{i,\ell} = f_\ell(x^i_{\text{qu}}, c),\ s_\ell(x^i_{\text{qu}}, c) = \mathcal{E}_\tau\left(\hat{y}^c_{i,\ell}, y^i_{\text{qu}}\right), \tag{5}$$

where $f_\ell$ is the target LLM and $\mathcal{E}_\tau$ is the evaluation metric (e.g., BLEU (Papineni et al., 2002) for machine translation task) associated with task $\tau$. These performance scores provide direct supervision for learning an ordering predictor, encouraging candidate demonstration orderings that yield better downstream task performance to be ranked better.

### 2.3 Ordering Predictor Training

We train the ordering predictor to estimate the performance score of a candidate demonstration ordering using the supervision constructed in Section 2.2. Specifically, we regress the observed performance score $s_\ell(x^i_{\text{qu}}, c)$ for each candidate demonstration ordering $c$ with a standard regression loss function. Let $\mathcal{S}_\ell = \left\{\left(x^i_{\text{qu}}, c, s_\ell(x^i_{\text{qu}}, c)\right)\right\}$ denote the set of query–ordering–score triples whose labels are generated by target LLM $f_\ell$. We train the predictor using mean squared error:

$$\mathcal{L}(\theta_\ell) = \frac{1}{|\mathcal{S}_\ell|} \sum_{(x,c,s)\in\mathcal{S}_\ell} \left(P_{\theta_\ell}(x, c) - s\right)^2. \tag{6}$$

We introduce two implementations of the ordering predictor $P_{\theta_\ell}$: ADOICLS and ADOICLE. ADOICLS represents each candidate ordering by encoding the complete ordered demonstration sequence, whereas ADOICLE constructs its representation by concatenating cached demonstration embeddings in the corresponding order. The two implementations share the same offline supervision and regression objective but use different candidate-order representations. ❶ **ADOICLS: Sequence-Level Ordering Predictor.** Given a training query $x^i_{\text{qu}} \in \boldsymbol{D}_{\text{qu}}$ and a sampled ordering $c = \pi(\mathcal{N}_k(x^i_{\text{qu}})) \in \tilde{\mathcal{C}}(x^i_{\text{qu}})$, ADOICLS concatenates the input–label pairs in $c$ in their specified order to form a complete in-context sequence. We encode the query and this ordered context separately as $\boldsymbol{v}^i_{\text{qu}} = f_{\text{e}}(x^i_{\text{qu}})$ and $\boldsymbol{v}^{\text{Seq}}_c = f_{\text{e}}(c)$. Given these two embeddings, the ordering predictor $P^{\text{Seq}}_{\theta_\ell}$ estimates the task performance of the target LLM $f_\ell$ on query $x^i_{\text{qu}}$ when using the demonstrations in order $c$:

$$\hat{s}^{\text{Seq}}_\ell(x^i_{\text{qu}}, c) = P^{\text{Seq}}_{\theta_\ell}\left(\boldsymbol{v}^i_{\text{qu}}, \boldsymbol{v}^{\text{Seq}}_c\right). \tag{7}$$

This formulation requires a separate context-encoding pass for each candidate ordering. ❷ **ADOICLE: Embedding-First Ordering Predictor.** Given the supervision constructed in Section 2.2, we adopt an embedding-first formulation for both training and inference. Specifically, for each query $x^i_{\text{qu}}$ in the query set $\boldsymbol{D}_{\text{qu}}$, its retrieved demonstration set in Equation (2) is $\mathcal{N}_k(x^i_{\text{qu}}) = \{(x^{ij}_{\text{in}}, y^{ij}_{\text{in}})\}^k_{j=1}$. We compute the query embedding once as $\boldsymbol{v}^i_{\text{qu}} = f_{\text{e}}(x^i_{\text{qu}})$, and compute a single joint embedding for each retrieved demonstration $\boldsymbol{v}^{ij}_{\text{demo}} = f_{\text{e}}\big(g(x^{ij}_{\text{in}}, y^{ij}_{\text{in}})\big)$, where $g(\cdot)$ denotes a predefined template that concatenates the input text and its corresponding label. Given a **sampled demonstration ordering** $c = \pi(\mathcal{N}_k(x^i_{\text{qu}})) \in \tilde{\mathcal{C}}(x^i_{\text{qu}})$, we construct its ordered in-context embedding by concatenating the corresponding demonstration embeddings:

$$\boldsymbol{v}_c = \text{Concat}\left(\boldsymbol{v}^{i\pi(1)}_{\text{demo}}, \boldsymbol{v}^{i\pi(2)}_{\text{demo}}, \ldots, \boldsymbol{v}^{i\pi(k)}_{\text{demo}}\right). \tag{8}$$

The embedding-first ordering predictor $P^{\text{Emb}}_{\theta_\ell}$ estimates the target-LLM performance score of candidate ordering $c$ as

$$\hat{s}^{\text{Emb}}_\ell(x^i_{\text{qu}}, c) = P^{\text{Emb}}_{\theta_\ell}\left(\boldsymbol{v}^i_{\text{qu}}, \boldsymbol{v}_c\right). \tag{9}$$

ADOICLE reuses cached query and per-demonstration embeddings across all candidate orderings for each query, avoiding repeated demonstration encoding while retaining explicit candidate-order construction and predictor scoring. When evaluating all $k!$ orderings, ADOICLE reuses cached embeddings to reduce demonstration-side encodings from $k!$ to $k$.

### 2.4 Test-Time Demonstration Ordering Selection

We illustrate the demonstration-order selection process using ADOICLE as an example; the corresponding procedure for ADOICLS is presented in the appendix. ADOICLE avoids repeated context encoding across candidate orderings by encoding the query once and reusing cached demonstration embeddings, reducing demonstration-side encoding passes from $k!$ to $k$ under exhaustive

search. All $k!$ candidate orderings are still scored, but without re-encoding their demonstration texts. Given a query $x^i_{\text{te}}$ from the test set, which is disjoint from the retrieval dataset, query set, and in-context set used for predictor training, we retrieve its top-$k$ demonstrations from the retrieval dataset $\boldsymbol{D}_{\text{re}}$, $\mathcal{N}_k(x^i_{\text{te}}) = \{(x^{ij}_{\text{re}}, y^{ij}_{\text{re}})\}_{j=1}^{k}$. We compute the query embedding $\boldsymbol{v}^i_{\text{te}} = f_{\text{e}}(x^i_{\text{te}})$ and the per-demonstration embeddings $\boldsymbol{v}^{ij}_{\text{demo}}$ using the same templates as in Section 2.3 (Equation (8)). For any candidate ordering $c = \pi(\mathcal{N}_k(x^i_{\text{te}}))$, its ordered in-context embedding $\boldsymbol{v}_c$ is constructed as in Equation (8), and its predicted effectiveness is obtained by the ordering predictor as in Equation (9): $\hat{s}^{\text{Emb}}_{\ell}(x^i_{\text{te}}, c) = P^{\text{Emb}}_{\theta_\ell}(\boldsymbol{v}^i_{\text{te}}, \boldsymbol{v}_c)$. We then select the ordering $c^*$ with the optimal predicted score:

$$c^* = \arg \underset{c \in \mathcal{C}_{\text{te}}(x^i_{\text{te}})}{\text{opt}} \hat{s}_\ell(x^i_{\text{te}}, c), \tag{10}$$

where $\hat{s}_\ell$ denotes either $\hat{s}^{\text{Seq}}_{\ell}$ or $\hat{s}^{\text{Emb}}_{\ell}$, and opt denotes $\arg\max$ for higher-is-better metrics and $\arg\min$ for error-based metrics. Finally, we construct the ICL prompt using $c^*$ and obtain the prediction $\hat{y}^i_{\text{te},\ell} = f_\ell(x^i_{\text{te}}, c^*)$ from the target LLM $f_\ell$.

**ADOICL Variants:** We further adapt ADOICL to visual tasks, many-shot demonstration settings, and few-shot predictor training. (1) **ADOICL for Vision Tasks.** We extend ADOICL to vision tasks by using the CLIP (Radford et al., 2021) model as the encoder: for each visual demonstration, we encode the image and the (text) label separately and concatenate them as the demonstration embedding, $\boldsymbol{v}^{ij}_{\text{demo}} = \text{Concat}(f_{\text{img}}(x^{ij}_{\text{re}}),\ f_{\text{text}}(y^{ij}_{\text{re}}))$. Different demonstration orderings are represented by concatenating the corresponding demonstration embeddings, and the ordering predictor is used to score candidate orders; all other components remain unchanged. (2) **Many-Shot Demonstration ADOICL.** For a large demonstration budget $k$, we cluster the $k$ demonstrations into a small number of groups $G$ (small number) and apply ADOICL at the group level, reducing the ordering space from $k!$ to $G!$. (3) **Few-Shot Training-Data ADOICL.** With limited training data, we replace sampling with multiple retrieval methods (e.g., BM25 and CR) to construct additional candidate demonstration sets. Despite limited training data, this approach yields diverse demonstration orderings for training the ordering predictor.

# 3 Experiments

## 3.1 Experimental Setup

**(1) Tasks, Datasets, and Metrics.** We evaluate ADOICL on five tasks using eight datasets. Specifically, text classification uses SST5 and Emotion with Accuracy metric; semantic textual similarity uses STS14 and STSB with MSE metric; summarization uses Gigaword with ROUGE-1 (Lin, 2004) metric; text expansion uses Gigatiny with ROUGE-1; and machine translation uses WMT19 En→Zh and Zh→En with BLEU metric. Higher Accuracy, ROUGE-1, and BLEU are better, while lower MSE is better. Details about tasks, datasets, and metrics are in Appendix Sections B, C and D. **(2) Target LLMs and Baselines.** We mainly use GLM 4-9B (GLM), LLaMA 3.1-8B (LLaMA), and Qwen2.5-7B (Qwen), and further evaluate scalability on 1B–70B models from different families, including LLaMA 3.2-1B, Qwen3.5 (4B/9B) (Team, 2026), DeepSeek-32B, LLaMA 3.3-70B, and GPT-4o; LLM URLs are listed in Appendix Table 8. We compare ADOICL with representative ICL baselines, including BM25 (Robertson et al., 2009), CR (Li & Qiu, 2023b), CD (Naik et al., 2023), and Kate (Liu et al., 2022); training-based methods, including TTF (Liu et al., 2025), PLR Batorski & Swoboda (2026), and DKNN (Li et al., 2025a); and ordering approaches, including ICCL (Liu et al., 2024), CDS Chung et al. (2026), and GlobalE (Lu et al., 2022). **(3) Other Setup.** We use BERT for text encoding and a 6-layer Transformer ordering predictor (Appendix Section E), trained with AdamW, a learning rate of $10^{-3}$, linear warmup, and prediction-head dropout ($p = 0.3$) to mitigate overfitting. We primarily use $k = 4$ demonstrations and sample $M = 5$ candidate orders per training query for offline supervision. In the main $k = 4$ setting, the predictor scores all $4!$ orderings at test time, and the target LLM is queried once with the selected ordering. All experiments run on NVIDIA A100 GPUs. The demonstration pool, predictor-training set, and test set are disjoint, with the test set reserved exclusively for final evaluation.

Table 1: Main results of ADOICL. Best/second-best results are in **bold**/underline. †: methods requiring training or a learned predictor; ∗: demonstration-ordering methods. Dashes (-) indicate that STS results are not reported for DKNN and TTF because they do not apply to regression.

| LLMs | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | Accuracy (%) ↑ | | | Accuracy (%) ↑ | | | ROUGE-1 ↑ | | | ROUGE-1 ↑ | | |
| Method | Classification: Emotion | | | Classification: SST5 | | | Summarization: Gigaword | | | Expansion: Gigatiny | | |
| BM25 | 54.3 | 28.3 | 60.3 | 34.2 | 29.4 | 50.0 | 0.028 | 0.020 | 0.023 | 0.119 | 0.121 | 0.126 |
| CR | 41.6 | 39.3 | 47.9 | 48.3 | 34.4 | 46.2 | 0.040 | 0.052 | 0.124 | 0.122 | 0.102 | 0.235 |
| CD | 55.8 | 37.0 | 53.8 | 45.0 | 34.2 | 45.7 | 0.032 | 0.056 | 0.151 | 0.117 | 0.105 | 0.298 |
| Kate | 56.1 | 44.8 | 59.3 | 46.2 | 34.4 | 48.9 | 0.056 | 0.050 | 0.120 | 0.197 | 0.129 | 0.297 |
| PLR†,∗ | 55.6 | 43.9 | 58.7 | 45.7 | 33.9 | 48.2 | 0.052 | 0.048 | 0.116 | 0.189 | 0.126 | 0.289 |
| CDS∗ | 55.3 | 43.5 | 58.4 | 45.4 | 33.7 | 47.8 | 0.051 | 0.047 | 0.114 | 0.187 | 0.124 | 0.286 |
| DKNN† | 60.8 | 45.3 | 54.0 | 48.0 | 36.7 | 44.1 | 0.058 | 0.056 | 0.158 | 0.201 | 0.116 | 0.256 |
| TTF† | 60.5 | 48.0 | 58.3 | 47.5 | 37.8 | 46.2 | 0.059 | 0.061 | 0.150 | 0.199 | 0.100 | 0.282 |
| ICCL∗ | 46.5 | 38.3 | 45.5 | 48.4 | 35.5 | 42.1 | 0.033 | 0.039 | 0.097 | 0.200 | 0.011 | 0.231 |
| GlobalE∗ | 50.3 | 45.0 | 44.8 | 39.6 | 33.7 | 45.0 | 0.030 | 0.028 | 0.130 | 0.146 | 0.117 | 0.203 |
| ADOICLE†,∗ | 70.8 | 57.5 | **69.0** | 51.6 | 39.9 | 52.3 | 0.065 | **0.081** | 0.179 | 0.235 | 0.133 | 0.341 |
| ADOICLS†,∗ | **71.4** | **58.1** | 68.6 | **52.0** | **40.3** | **53.8** | **0.069** | 0.078 | **0.183** | **0.239** | **0.136** | **0.347** |
| Metric | MSE ↓ | | | MSE ↓ | | | BLEU ↑ | | | BLEU ↑ | | |
| Task & Data | STS:STS14 | | | STS:STSB | | | Translation: WMT19 En-Zh | | | Translation: WMT19 Zh-En | | |
| BM25 | 1.613 | 6.433 | 1.549 | 1.536 | 4.030 | 1.549 | 0.034 | 0.018 | 0.040 | 0.327 | 0.081 | 0.040 |
| CR | 11.812 | 11.630 | 8.291 | 8.291 | 8.079 | 8.291 | 0.015 | 0.015 | 0.016 | 0.199 | 0.087 | 0.066 |
| CD | 0.826 | 6.075 | 0.802 | 0.788 | 2.194 | 0.802 | 0.214 | 0.022 | 0.120 | 0.424 | 0.078 | 0.468 |
| Kate | 0.797 | 6.038 | 0.685 | 0.787 | 1.638 | 0.685 | 0.201 | 0.021 | 0.130 | 0.466 | 0.091 | 0.514 |
| PLR†,∗ | 0.821 | 6.124 | 0.704 | 0.805 | 1.705 | 0.701 | 0.196 | 0.020 | 0.126 | 0.452 | 0.089 | 0.501 |
| CDS∗ | 0.836 | 6.158 | 0.717 | 0.817 | 1.732 | 0.713 | 0.193 | 0.019 | 0.124 | 0.446 | 0.087 | 0.493 |
| DKNN† | — | — | — | — | — | — | 0.205 | 0.022 | 0.129 | 0.469 | 0.074 | 0.528 |
| TTF† | — | — | — | — | — | — | **0.233** | 0.022 | 0.097 | 0.463 | 0.091 | 0.519 |
| ICCL∗ | 1.182 | 6.075 | 0.796 | 0.846 | 1.770 | 1.153 | 0.213 | 0.025 | 0.125 | 0.305 | 0.092 | 0.280 |
| GlobalE∗ | 0.813 | 6.952 | 0.872 | 0.934 | 4.312 | 0.690 | 0.216 | 0.015 | 0.041 | 0.376 | 0.083 | 0.142 |
| ADOICLE†,∗ | 0.768 | 5.871 | **0.645** | 0.752 | **1.486** | 0.654 | 0.232 | 0.026 | 0.136 | 0.495 | 0.095 | 0.565 |
| ADOICLS†,∗ | **0.752** | **5.830** | 0.671 | **0.739** | 1.510 | **0.641** | 0.231 | **0.029** | **0.139** | **0.512** | **0.098** | **0.582** |

Table 2: Ablation results for ADOICLE. ADOICLS ablation results are in Appendix Section H.

| Ablation setting | | | Results | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Similarity-based retrieval | Ordering predictor | Dual-input architecture | Accuracy (%) ↑ Classification: Emotion | | | Accuracy (%) ↑ Classification: SST5 | | | ROUGE-1 ↑ Summarization: Gigaword | | | ROUGE-1 ↑ Expansion: Gigatiny | | |
| | | | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ✓ | ✓ | ✓ | **70.8** | **57.5** | **69.0** | **51.6** | **39.9** | **52.3** | **0.065** | **0.081** | **0.179** | **0.235** | **0.133** | **0.341** |
| × | ✓ | ✓ | 60.5 | 46.3 | 59.1 | 50.6 | 38.5 | 50.2 | 0.060 | 0.066 | 0.176 | 0.207 | 0.131 | 0.327 |
| ✓ | × | ✓ | 56.1 | 44.8 | 59.3 | 46.2 | 34.4 | 48.9 | 0.056 | 0.050 | 0.120 | 0.197 | 0.129 | 0.297 |
| ✓ | ✓ | × | 65.5 | 48.0 | 67.3 | 48.6 | 37.1 | 47.1 | 0.061 | 0.063 | 0.160 | 0.215 | 0.126 | 0.276 |

### 3.2 Main Results and Ablation Study

**Main results.** As shown in Table 1, both ADOICLE and ADOICLS achieve consistently strong performance across datasets and target LLMs. ADOICLS is overall the strongest variant, obtaining the best result in most settings, e.g., 71.4% classification accuracy. ADOICLE, our main embedding-first implementation, is also highly competitive and often ranks second, e.g., 0.179 ROUGE-1 on summarization. Across nearly all settings, the two ADOICL variants occupy the best or second-best positions; the only exception is WMT19 En-Zh with GLM 4-9B, where TTF ranks first by only 0.001 BLEU over ADOICLE. Overall, these results show that performance-oriented demonstration ordering provides robust gains over prior ICL methods. **Ordering quality and oracle gap.** We all evaluate all $4! = 24$ permutations of each query's fixed demonstration set, comparing ADOICL against original, reverse, random, and best fixed orderings and a per-query oracle (Appendix Section L, Appendix Table 24). ADOICLS/ADOICLE outperform all four baselines in 12/11 of 12 settings. On Emotion with Llama, ADOICLS raises accuracy from 42.0% to 58.1%, shrinking the oracle gap from 20.7 to 4.6 points. These results support query-dependent ordering selection while indicating room for further improvement. **Performance across repeated runs.** On SST5, both ADOICL variants outperform all baselines in mean accuracy over three runs across all three LLMs. ADOICLS improves over the strongest baseline by 2.5–3.8 percentage points, with standard devi-

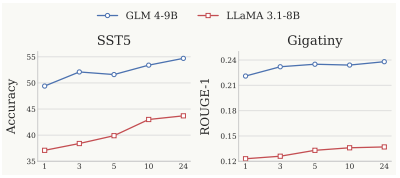


Figure 3: Results with different sampled order numbers ($M$).

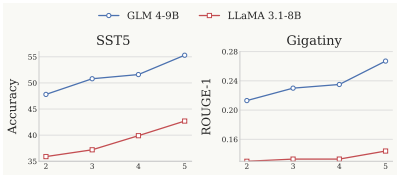


Figure 4: Results with different demonstration numbers ($k$).

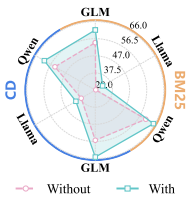


Figure 5: Plug-and-play results on Emotion.

ations comparable to those of the baselines. See Appendix Section J and Appendix Table 23 for details.

**Ablation study.** We primarily report ADOICLE ablations (Table 2 and Appendix Table 22); ADOICLS ablations are in Appendix Section H. (I) **Similarity-based retrieval.** Similarity-based retrieval outperforms diversity-based retrieval overall, improving average accuracy by 5.98% and reducing MSE by 0.112 across target LLMs; we therefore use it as ADOICL's default. (II) **Ordering predictor.** Removing the predictor consistently degrades performance, with an average accuracy drop of 8.57% and an MSE increase of 0.076, confirming that ordering-performance prediction is crucial for identifying high-performing demonstration orderings. (III) **Dual-input architecture.** The dual-input architecture consistently outperforms the single-input variant, yielding 4.58% higher accuracy and an average MSE reduction of 0.134, confirming the benefit of separately embedding the query and contextual demonstrations for ordering-performance prediction. (IV) **Effect of sampled candidate orders $M$.** Figure 3 studies the number of sampled candidate orders used for offline supervision. Here, $M$ affects training-data construction only; in the main $k = 4$ setting, test-time scoring uses the full $4!$ candidate pool. Increasing $M$ generally improves or stabilizes performance by providing richer supervision, but also increases the cost of constructing training data. Thus, $M = 5$ offers a practical cost-quality trade-off. (V) **Effect of demonstration number $k$.** Figure 4 shows that ADOICLE generally improves from $k = 2$ to $k = 4$, indicating that the predictor benefits from more candidate demonstration orderings while remaining effective across different $k$. (VI) **Effect of training-set construction.** Appendix Table 13 shows that ADOICLE remains competitive under random, stratified, and cluster-based leakage-free splits with the same query/in-context ratio, indicating that its performance is not tied to a specific random split.

## 4 Generalization and Discussion

**Generalization.** We focus on ADOICLE generalization; ADOICLS results appear in Appendix Section I. **(1) Cross-LLM generalization.** The ordering predictor transfers across LLMs without retraining, with slightly lower performance than target-specific training (Table 3); e.g., training on GLM 4-9B yields 51.5% accuracy on Llama 3.1-8B. **(2) Cross-dataset generalization.** We train the ADOICL *ordering predictor* on SST5 and evaluate it on Emotion. As Appendix Table 25 shows, the transferred predictor achieves 58.6% average accuracy, outperforming the baselines in Table 2, indicating that it learns ordering signals transferable across datasets. **(3) Cross-task generalization.** The ordering predictor also transfers across tasks: trained on classification data (Emotion), it achieves 0.094 average ROUGE-1 on summarization (Gigaword), outperforming all baselines (Table 4). We argue that summarization, while generative in form, is governed by sequential micro-decisions whose structure aligns with classification-style inference. Under this perspective, a classification-trained predictor can remain effective when applied to summarization. **(4) Cross-demonstration-number generalization.** We vary the demonstration number $k$ while fixing the predictor's input arity to $w = 4$. To accommodate this mismatch, we split demonstrations into text segments when $k < w$ and merge adjacent demonstrations into segments when $k > w$, before encoding and pooling them into four input vectors. On GLM-4-9B, ADOICLE achieves 69.2/70.8/73.3% accuracy on Emotion and 0.064/0.065/0.069 ROUGE-1 on Gigaword for $k = 3/4/5$, respectively. These results show that the fixed-input predictor can accommodate different demonstration numbers through segment-based adaptation. See Appendix Section F for details.

**Discussion.** *We mainly conduct the discussion on ADOICLE.* (1) **Plug-and-play.** As shown in Figure 5, Appendix Figure 6, and Appendix Table 10, ADOICLE's *ordering predictor* can be plugged into BM25- or CD-based ICL pipelines by re-ranking their retrieved demonstrations. It consistently improves average accuracy from 48.25% to 53.95% and average ROUGE-1 from 0.052 to 0.055

Table 3: Results of cross-LLM generalization. **Training LLM**: the LLM used to provide performance scores for constructing ADOICL training data. **Testing LLM**: the LLM used to construct the in-context examples during inference.

| Dataset & Metric | Emotion & Accuracy (%) ↑ | | | Gigaword & ROUGE-1 ↑ | | |
|---|---|---|---|---|---|---|
| Testing LLM | Llama | | | Llama | | |
| Training LLM | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ADOICL ↑ | 51.5 | 57.5 | 47.9 | 0.062 | 0.081 | 0.063 |

Table 4: The results of cross-task generalization. The ADOICLE ordering predictor is trained on the Emotion for the classification task and evaluated on the Gigaword for the summarization task.

| Training Task / Dataset | Classification / Emotion | | |
|---|---|---|---|
| Testing Task / Dataset | Summarization / Gigaword | | |
| Target LLMs | GLM | Llama | Qwen |
| ROUGE-1 ↑ | 0.056 | 0.065 | 0.161 |

without retraining the *ordering predictor*, suggesting that it captures performance-relevant ordering signals. (2) **Results on more target LLM families.** We further evaluate ADOICLE on five additional target LLMs (1B–70B) and GPT-4o. Appendix Table 11 shows strong results across model families, reaching up to 79.0% accuracy and reducing MSE to 0.88. On GPT-4o ( Table 6), ADOICLE achieves 63.9% accuracy; even when the ordering predictor is trained only on GLM4-9B, it still attains 59.8% on GPT-4o. This cross-LLM setup reduces data-construction cost by training on open GLM4-9B instead of paid GPT-4o, yet the ordering predictor remains effective when deployed on larger commercial LLMs. Overall, ADOICLE generalizes well across LLM scales, families, and commercial LLMs.

(3) **Few-shot training texts.** With only 30 training texts, we use multiple retrievers, including BM25, CD, CR, and similarity-based retrieval, to construct diverse candidate demonstration sets. Although the number of texts is small , each text yields many distinct text–in-context pairs and induced orderings, providing ample supervision for the ordering predictor. As shown in Appendix Table 21, ADOICLE still achieves up to 62.8% accuracy. (4)**ADOICL Variants and Vision Generalization.** ADOICL supports broader settings with minor adaptations: for vision tasks, it uses CLIP to encode each image–text demonstration as $\mathrm{Concat}(f_{\text{img}}(x), f_{\text{text}}(y))$. On CIFAR-10 image classification with InternVL 2.5 (4B/8B/26B) and $k$=4 demonstrations, we compare ADOICL with image-applicable baselines, including CD, CR, KATE, and ICL-rerank (Zhou et al., 2024). As shown in Appendix Table 20, ADOICL achieves the best performance, reaching **99.3%** accuracy and demonstrating its effectiveness for visual ICL. (5)**100-shot demonstration setting.** We further evaluate ADOICL with $k$=100 demonstrations. Appendix Table 12 shows that ADOICLE remains SOTA, achieving up to 78.1% accuracy and 0.54 ROUGE-1. (6) **Cost and time.** For offline supervision, ADOICLE evaluates only $M = 5$ sampled candidate orders per training query rather than all $k!$ permutations. On SST5, constructing training data with GPT-4o costs 4.7397 USD, while using an open-source LLM such as GLM4-9B incurs no API charges and takes about 0.28 hours with local computation. Training the ordering predictor takes 1.4 hours. At inference, ADOICLE costs 0.24 seconds per query, including retrieval, predictor scoring, and final target-LLM inference; the predictor itself takes only 0.0074 seconds. In Table 5, ADOICLE is only 0.02 seconds slower than the fastest baseline while improving average accuracy by 5%, and the one-time costs are amortized over repeated deployments.

Table 5: Average testing time on SST5 across three target LLMs, reported as the mean over the three models. Time is measured in seconds per text. The average accuracy and average latency are computed by averaging over the three LLMs.

| Method | BM25 | CR | CD | Kate | DKNN | TTF | ICCL | GlobalE | ADOICL |
|---|---|---|---|---|---|---|---|---|---|
| Accuracy (%) | 37.9 | 43.0 | 41.6 | 43.1 | 42.9 | <u>44.1</u> | 42.0 | 39.4 | **47.9** |
| Time (s) | 1.65 | 0.27 | <u>0.23</u> | <u>0.23</u> | **0.22** | <u>0.23</u> | 0.31 | 0.32 | 0.24 |

Table 6: SST-5 Results on GPT-4o: ADOICL denotes GPT-4o training, while ADOICLE (GLM) denotes training on GLM4-9B and testing on GPT-4o.

| Method | BM25 | CR | CD | Kate | DKNN | TTF | ICCL | GlobalE | ADOICLE | ADOICLE (GLM) |
|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy (%) ↑ | 52.1 | 51.6 | 52.4 | 56.0 | 57.6 | 58.1 | 55.1 | 56.2 | 63.9 | 59.8 |

## 5 Conclusion

We find that demonstration ordering should be treated as an adaptive and performance-oriented inference-time decision, conditioned on the input query, retrieved demonstrations, and target LLM. We propose ADOICL, a performance-prediction framework that estimates the downstream performance of candidate demonstration orderings and selects the most promising ordering before a single target-LLM inference. ADOICL also demonstrates plug-and-play compatibility, visual ICL support, few-shot training-data effectiveness, and a clustering-based group-level scalability variant for many-shot settings.

### AI USE STATEMENT

In this work, we used generative AI tools to improve the grammar, clarity, and presentation of the manuscript, and to assist with writing and debugging portions of the implementation code. The research problem, core methodological ideas, experimental design, evaluation protocol, and interpretation of the results were developed by the authors.

All AI-assisted text was reviewed and revised by the authors. All AI-assisted code was manually inspected, integrated into the codebase, and tested by the authors for consistency with the intended algorithms and experimental settings. The authors independently executed the experiments and verified the reported results. We did not use generative AI to fabricate or alter data, experimental results, citations, or scientific claims.

We take responsibility for the final content of this work, including all text, code, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work does not involve human participants or the collection of personal or sensitive information. All experiments use established public benchmarks and existing open-source or commercial LLMs in accordance with their applicable licenses and terms. The proposed method may inherit biases and undesirable behaviors from these datasets and models and should therefore be used cautiously in high-stakes applications. The authors take full responsibility for the research presented in this work.

### REPRODUCIBILITY STATEMENT

All mathematical formulations, training-data construction procedures, ordering-predictor architectures, training objectives, and test-time inference procedures are detailed in the Method section and Appendix Sections E. Complete evaluation protocols, metric definitions, dataset sources, benchmark splits, and implementation settings are documented in the Experimental Setup section and Appendix Sections B–D. The full source code, model checkpoints, configuration files, and evaluation scripts will be made publicly available upon paper acceptance.

## REFERENCES

Pawel Batorski and Paul Swoboda. Plr: Plackett-luce for reordering in-context learning examples. *arXiv preprint arXiv:2603.21373*, 2026.

Rahul Atul Bhope, Praveen Venkateswaran, KR Jayaram, Vatche Isahagian, Vinod Muthusamy, and Nalini Venkatasubramanian. Optiseq: Optimizing example ordering for in-context learning. *CoRR*, 2025.

Samuel Camba, Abraham Otero, Daniel García, Luciano Sánchez, and Nahuel Costa. Can in-context learning enable large vision language models to detect ecg abnormalities? In *Conference of the Spanish Society of Artificial Intelligence in Biomedicine*, pp. 200–213. Springer, 2025.

Ioannis Charisiadis, Ilyass El Allali, Richard GP Lopata, Clara I Sánchez, and Navchetan Awasthi. Icl-noiseunet-a novel in-context learning based framework for ultrasound segmentation with adaptive noise modulation. In *Medical Imaging with Deep Learning*, 2026.

Peng Chen, Wenxuan He, Feng Qian, Guangyao Shi, and Jingwen Yan. A synergistic cnn-transformer network with pooling attention fusion for hyperspectral image classification. *Digital Signal Processing*, 160:105070, 2025.

Tsz Ting Chung, Lemao Liu, Mo Yu, and Dit-Yan Yeung. Many-shot cot-icl: Making in-context learning truly learn. *arXiv preprint arXiv:2605.13511*, 2026.

Joy Crosbie and Ekaterina Shutova. Induction heads as an essential mechanism for pattern matching in in-context learning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5049–5111, 2025.

Lizhe Fang, Yifei Wang, Khashayar Gatmiry, Lei Fang, and Yisen Wang. Rethinking invariance in in-context learning. *arXiv preprint arXiv:2505.04994*, 2025.

Qi Guo, Leiyu Wang, Yidong Wang, Wei Ye, and Shikun Zhang. What makes a good order of examples in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 14892–14904, 2024.

Pengfei He, Yingqian Cui, Han Xu, Hui Liu, Makoto Yamada, Jiliang Tang, and Yue Xing. Towards the effect of examples on in-context learning: A theoretical case study. *Stat*, 14(1):e70045, 2025.

Yinhan He, Wendy Zheng, Song Wang, Zaiyi Zheng, Yushun Dong, Yaochen Zhu, and Jundong Li. Hierarchical demonstration order optimization for many-shot in-context learning. *Advances in Neural Information Processing Systems*, 38:92325–92356, 2026.

Zhonglin Jiang, Shenghan Gao, Qian Tang, Zequn Wang, Yu Liu, and Hong-Zhong Huang. Generative reliability-based design optimization using in-context learning capabilities of large language models. *Journal of mechanical design*, 148(2):021713, 2026.

Chuyuan Li, Raymond Li, Thalia S. Field, and Giuseppe Carenini. Delta-KNN: Improving demonstration selection in in-context learning for Alzheimer's disease detection. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25807–25826, Vienna, Austria, July 2025a. Association for Computational Linguistics.

Warren Li, Yiqian Wang, Zihan Wang, and Jingbo Shang. Order matters: Rethinking prompt construction in in-context learning. *arXiv preprint arXiv:2511.09700*, 2025b.

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

Xiaonan Li and Xipeng Qiu. Mot: Memory-of-thought enables chatgpt to self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6354–6374, 2023b.

Yanshu Li, Jianjiang Yang, Tian Yun, Pinyuan Feng, Jinfa Huang, and Ruixiang Tang. Taco: Enhancing multimodal in-context learning via task mapping-guided sequence configuration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 736–763, 2025c.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Hui Liu, Wenya Wang, Hao Sun, Chris Xing Tian, Chenqi Kong, Xin Luna Dong, and Haoliang Li. Unraveling the mechanics of learning-based demonstration selection for in-context learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2623–2641, 2025.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114, 2022.

Renpu Liu and Jing Yang. Unlabeled data can provably enhance in-context learning of transformers. *Advances in Neural Information Processing Systems*, 38:91895–91932, 2026.

Yinpeng Liu, Jiawei Liu, Xiang Shi, Qikai Cheng, Yong Huang, and Wei Lu. Let's learn step by step: Enhancing in-context learning ability with curriculum learning. *arXiv preprint arXiv:2402.10738*, 2024.

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 *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Yue M Lu, Mary Letey, Jacob A Zavatone-Veth, Anindita Maiti, and Cengiz Pehlevan. Asymptotic theory of in-context learning by linear attention. *Proceedings of the National Academy of Sciences*, 122(28):e2502599122, 2025.

Antonio Moreno-Ortiz and María García-Gámez. Emotion in context: Human and ai perspectives on the affective landscape of severance. *Corpus Pragmatics*, 10(1):15, 2026.

Ranjita Naik, Varun Chandrasekaran, Mert Yuksekgonul, Hamid Palangi, and Besmira Nushi. Diversity of thought improves reasoning abilities of large language models. 2023.

Tai Nguyen and Eric Wong. In-context example selection with influences. *arXiv preprint arXiv:2302.11042*, 2023.

Josef Ott. Context collapse: In-context learning and model collapse. *arXiv preprint arXiv:2601.00923*, 2026.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Chengwei Qin, Aston Zhang, Chen Chen, Anirudh Dagar, and Wenming Ye. In-context learning with iterative demonstration selection. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7441–7455, 2024.

Jingang Qu, David HolzmÃžller, GaÃl Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2655–2671, Seattle, United States, 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.naacl-main.191.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Hsuan Chen. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3687–3697, 2018.

Lida Shi, Fausto Giunchiglia, Ran Luo, Daqian Shi, Rui Song, Xiaolei Diao, and Hao Xu. An empirical study of llms via in-context learning for stance classification. *Information Processing & Management*, 63(1):104322, 2026.

Yanpeng Sun, Qiang Chen, Jian Wang, Jingdong Wang, and Zechao Li. Exploring effective factors for improving visual in-context learning. *IEEE Transactions on Image Processing*, 2025.

Eshaan Tanwar, Subhabrata Dutta, Manish Borthakur, and Tanmoy Chakraborty. Multilingual llms are better cross-lingual in-context learners with alignment. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 6292–6307. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.346. URL https://doi.org/10.18653/v1/2023.acl-long.346.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Jianyu Wang, Zhiqiang Hu, and Lidong Bing. Evolving prompts in-context: An open-ended, self-replicating perspective. *arXiv preprint arXiv:2506.17930*, 2025.

Song Wang, Zihan Chen, Chengshuai Shi, Cong Shen, and Jundong Li. Mixture of demonstrations for in-context learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 88091–88116. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a0da098e0031f58269efdcba40eedf47-Paper-Conference.pdf.

Wenhao Wang, Yifan Sun, Zhentao Tan, and Yi Yang. Anypattern: Towards in-context image copy detection. *International Journal of Computer Vision*, 134(2):80, 2026.

Yuxuan Wen, Xiaoming Chen, Maojun Zhang, Zhaohui Yang, Chongwen Huang, and Zhaoyang Zhang. Icwlm: A multi-task wireless large model via in-context learning. *IEEE Transactions on Communications*, 2026.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1423–1436, 2023.

Kayo Yin and Jacob Steinhardt. Which attention heads matter for in-context learning? *arXiv preprint arXiv:2502.14010*, 2025.

Hang Zhang, Yeyun Gong, Xingwei He, Dayiheng Liu, Daya Guo, Jiancheng Lv, and Jian Guo. Noisy pair corrector for dense retrieval. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11439–11451, 2023. doi: 10.18653/v1/2023.findings-emnlp.765. URL https://aclanthology.org/2023.findings-emnlp.765.

Yucheng Zhou, Xiang Li, Qianning Wang, and Jianbing Shen. Visual in-context learning for large vision-language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 15890–15902, 2024.

# APPENDIX

This appendix includes the following supplementary materials:

- Related work (Section A)
- Formal task definitions (Section B)
- Dataset details (Section C)
- Detailed definitions of the evaluation metrics (Section D)
- Ordering predictor architecture (Section E)
- Cross-demonstration-number generalization (Section F)
- Details of ADOICL variants (Section G)
- Ablation results for ADOICLS (Section H)
- Generalization results for ADOICLS (Section I)
- Performance across repeated runs (Section J)
- Limitations (Section K)
- Ordering quality with fixed demonstration sets (Section L)
- Comparison with other demonstration-ordering methods (Section M)
- Ordering selection quality on Gigatiny (Section N)

## A RELATED WORK

In-context learning (ICL) (Shi et al., 2026; Ott, 2026; Zhang et al., 2023; Wang et al., 2026; Fang et al., 2025; Li et al., 2025c) performance depends on several inference-time decisions, including which demonstrations to retrieve, how to transform them, and how to arrange them in the prompt. Prior work has investigated these decisions using a range of heuristic, information-theoretic, and learned criteria (Wen et al., 2026; Jiang et al., 2026; Liu & Yang, 2026; Charisiadis et al., 2026; Camba et al., 2025; Yin & Steinhardt, 2025; He et al., 2025; Crosbie & Shutova, 2025). Our work focuses on predicting the target LLM's task performance for candidate permutations of a retrieved demonstration set, conditioned on the input query.

**Demonstration Retrieval and Transformation.** A large body of work studies how to select or construct effective demonstrations for ICL. Early unsupervised methods rely on similarity-based heuristics such as top-$k$ nearest neighbors (Wang et al., 2025; Tanwar et al., 2023; Lu et al., 2025), later enhanced with perplexity, output probabilities, or information-theoretic criteria (Sun et al., 2025; Chen et al., 2025; Nguyen & Wong, 2023; Li & Qiu, 2023a). Supervised approaches learn selection policies via re-ranking (Rubin et al., 2022), multi-task retrieval, or reinforcement learning. Beyond retrieval, prior work also explores demonstration transformation, including synthetic example generation, latent feature manipulation, and compositional context construction (Wang et al., 2024). PDDS predicts task performance to select demonstration sets, whereas our method predicts the effectiveness of candidate permutations within a retrieved demonstration set.

**Demonstration Ordering.** Several studies explicitly examine demonstration ordering, motivated by the sensitivity of LLMs to input sequence (Qu et al., 2025). Existing approaches include both heuristic rules and adaptive ordering methods. Self-Adaptive ICL (Wu et al., 2023) introduces a select-then-rank framework that adapts demonstration selection and ordering to individual inputs. DEmO (Guo et al., 2024) filters candidate orders using label fairness and then selects an order for each test instance. OptiSeq (Bhope et al., 2025) performs inference-time search using output log probabilities to prune candidate orderings. PLR (Batorski & Swoboda, 2026) learns a Plackett–Luce distribution over permutations and updates it using task-level performance. These methods establish that demonstration ordering can be adaptive and directly informed by downstream outcomes.

Ordering has also been studied in many-shot settings. HIDO (He et al., 2026) combines an information-theoretic order evaluation criterion with hierarchical search. Curvilinear Demonstration Selection (CDS) (Chung et al., 2026) organizes demonstrations to encourage gradual conceptual progression in many-shot chain-of-thought ICL. These approaches provide relevant context for our group-based many-shot extension.

ADOICL investigates a query-conditioned regression approach to ordering: an offline-trained predictor estimates the task performance of each candidate permutation, and the highest-scoring candidate is selected at inference time. The embedding-first variant reuses demonstration representations across candidate permutations to reduce repeated encoding. This design differs from inference-time search based on target-model output probabilities and from optimizing a probability distribution over permutations. We evaluate the approach across classification, regression, and generation tasks, using task-specific performance supervision.

**Adaptive Decisions in ICL.** Beyond ordering, several works investigate adaptive inference-time decisions in ICL, most notably dynamically selecting the number of demonstrations rather than fixing $k$ (Sun et al., 2025; Qin et al., 2024). These studies address demonstration count, while Self-Adaptive ICL and DEmO also establish input-dependent selection and ordering. Our work builds on this broader perspective of adaptive context construction and focuses on selecting permutations through explicit prediction of task performance.

**Summary.** Prior work has advanced demonstration retrieval, transformation, adaptive ordering, and performance-based context selection. Within this literature, ADOICL focuses on learning an order-sensitive predictor for candidate permutations of a retrieved demonstration set. Its contribution lies in the specific query-conditioned ordering model, the reuse of demonstration embeddings during candidate evaluation, and the empirical evaluation across tasks and target LLMs.

# B TASK

We evaluate five tasks spanning language understanding and generation: text classification, textual similarity estimation (STS), abstractive summarization, text expansion, and translation.

## B.1 TEXT CLASSIFICATION

Text classification assigns a discrete label to an input text. Given a document $d$ and a label set $L = \{l_1, \ldots, l_k\}$, the goal is to learn a mapping function $f(d) \in L$ from labeled data. We evaluate this task primarily with accuracy.

## B.2 SEMANTIC TEXTUAL SIMILARITY

Semantic textual similarity measures how close in meaning two texts are. Given a pair $(t_1, t_2)$, the model predicts a real-valued similarity score (dataset-specific range), where larger values indicate higher semantic alignment. We evaluate this regression task using MSE.

## B.3 SUMMARIZATION

Summarization generates a concise text that captures the key information of an input document while allowing paraphrasing and synthesis beyond extractive copying. For an input document $D$, the model outputs a shorter summary $S$ that preserves the main content of $D$. We evaluate this task using ROUGE.

## B.4 TEXT EXPANSION

Text expansion produces a longer, coherent elaboration of a brief input while preserving its meaning. Given a short input $I$, the output $E$ should remain faithful to $I$ and add plausible details or explanations. We evaluate content coverage using ROUGE-style metrics.

## B.5 TRANSLATION

Machine translation converts text from a source language to a target language while preserving meaning and fluency. Given a source sentence $s_{\mathrm{src}}$ in language $L_{\mathrm{src}}$, the model generates a target sentence $s_{\mathrm{tgt}}$ in language $L_{\mathrm{tgt}}$. We report BLEU as a standard automatic evaluation metric.

# C DATASET

This appendix describes the datasets used in our experiments: SST5, Emotion, STS14, STSB, Gigaword, GigaTiny, and WMT19 En–Zh, WMT19 Zh–En. Together, they cover a broad set of NLP evaluations, including text classification, semantic textual similarity (STS), abstractive summarization, text expansion, and machine translation.

## C.1 SST5 (STANFORD SENTIMENT TREEBANK; 5-WAY)

SST5 is a fine-grained sentiment classification benchmark built from movie-review sentences. It contains 11,855 sentences and a much larger set of phrase-level constituents obtained from parse trees (215,154 phrases). Each sentence/phrase is annotated by human raters with one of five sentiment labels: *very negative*, *negative*, *neutral*, *positive*, and *very positive*. The fully labeled parse structures make SST5 suitable for studying compositional sentiment as well as standard 5-class text classification.

## C.2 EMOTION

Emotion is an English Twitter-based dataset for emotion recognition. Each tweet is labeled with one of six categories: *anger*, *fear*, *joy*, *love*, *sadness*, or *surprise*. It is commonly used as a multi-class classification benchmark and highlights the subtle, context-dependent nature of affective language in short social-media texts.

## C.3 STS14 (SEMANTIC TEXTUAL SIMILARITY 2014)

STS14 comes from SemEval-2014 Task 10 and evaluates semantic similarity estimation on sentence pairs drawn from multiple domains (e.g., news headlines, image captions, and forum posts). Each pair is annotated with a similarity score from 0 (unrelated) to 5 (semantically equivalent). The dataset is designed to test a model's ability to quantify meaning overlap between two sentences.

## C.4 STSB (SEMANTIC TEXTUAL SIMILARITY BENCHMARK)

STSB is a standardized benchmark that consolidates sentence pairs from SemEval STS tasks (2012–2016). The pairs cover diverse sources such as captions, news, and forums, and are labeled with continuous similarity scores (often normalized to $[0, 1]$ for evaluation). STSB is widely used to assess regression-based similarity models, including embedding- and transformer-based approaches, typically reporting correlation-based metrics.

## C.5 GIGAWORD

We use the English Gigaword newswire corpus as a benchmark for abstractive summarization, commonly instantiated as headline generation. The corpus contains millions of news articles collected over multiple years from major news agencies (e.g., AFP and AP). Summarization models are trained/evaluated on paired inputs (article text, typically lead sentences) and targets (headlines or reference summaries).

## C.6 GIGATINY

GigaTiny is a lightweight subset of Gigaword intended for fast iteration and resource-efficient experiments. While smaller in scale, it preserves the same news-style structure as the full corpus, making it convenient for preliminary studies in summarization and text expansion before moving to larger datasets.

## C.7 WMT19 EN–ZH (WORKSHOP ON MACHINE TRANSLATION 2019)

WMT19 En–Zh provides parallel data for English↔Chinese translation as part of the WMT 2019 shared tasks. It aggregates bilingual sentence pairs from multiple sources (e.g., Common Crawl and

News Commentary) at a large scale. The dataset is widely used to evaluate sequence-to-sequence translation systems, with BLEU being a standard automatic metric.

### C.8 WMT19 Zh–En (Workshop on Machine Translation 2019)

WMT19 Zh–En is the Chinese-to-English counterpart of WMT19 En–Zh in the WMT 2019 shared tasks. It uses the same large-scale parallel corpora collected from multiple sources (e.g., Common Crawl and News Commentary), but treats Chinese as the source language and English as the target language. This dataset is commonly used to evaluate sequence-to-sequence Zh→En translation systems, with BLEU as a standard automatic metric.

Table 7: Datasets and Their URLs

| **Task**<br>**Dataset**<br>**URL** |
|---|
| Textual Similarity Estimation |
| STSB<br>https://huggingface.co/datasets/SetFit/stsb |
| STS14<br>https://huggingface.co/datasets/mteb/sts14-sts |
| Text Classification |
| Emotion<br>https://huggingface.co/datasets/dair-ai/emotion |
| SST5<br>https://huggingface.co/datasets/SetFit/sst5 |
| Abstractive Summarization / Text Expansion |
| gigatiny<br>https://huggingface.co/datasets/SpeedOfMagic/gigaword_tiny |
| gigaword<br>https://huggingface.co/datasets/Harvard/gigaword |
| Translation |
| WMT19 En–Zh<br>https://huggingface.co/datasets/WillHeld/wmt19-valid-only-zh_en |

## D Metrics

This appendix defines the evaluation metrics used in our experiments: BLEU, ROUGE-1, Mean Squared Error (MSE), and accuracy. We use these metrics to assess LLM performance on translation, summarization/expansion, semantic textual similarity (STS), and text classification, respectively.

### D.1 BLEU (Bilingual Evaluation Understudy)

BLEU is a precision-oriented metric for machine translation. It computes the geometric mean of modified $n$-gram precisions and applies a brevity penalty to discourage overly short hypotheses. For a candidate translation $C$ and references $\{R_i\}$, BLEU is:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \tag{11}$$

Table 8: URLs for the LLMs.

| Model | URL |
|---|---|
| GPT-4o | https://platform.openai.com/docs/models/gpt-4o |
| GLM4 9B | https://huggingface.co/zai-org/glm-4-9b |
| Qwen2.5-7b | https://huggingface.co/Qwen/Qwen2.5-7B |
| Qwen3.5-4b | https://huggingface.co/Qwen/Qwen3.5-4B |
| Qwen3.5-9b | https://huggingface.co/Qwen/Qwen3.5-9B |
| LLAMA-3.1-8b | https://huggingface.co/meta-llama/Llama-3.1-8B |
| LLAMA-3.2-1b | https://huggingface.co/meta-llama/Llama-3.2-1B |
| LLAMA-3.2-3b | https://huggingface.co/meta-llama/Llama-3.2-3B |
| LLAMA-3.3-70b | https://huggingface.co/meta-llama/Llama-3.3-70B |

Table 9: Summary of Evaluation Tasks, Datasets, and Corresponding Metrics.

| Task Domain | Benchmark Datasets | Metric | Task Type |
|---|---|---|---|
| Text Classification | SST-5, Emotion Saravia et al. (2018) | Accuracy | Classification |
| Semantic Textual Similarity | STSB, STS14 | MSE | Regression |
| Summarization | Gigaword | ROUGE-1 | Generation |
| Text Expansion | Gigatiny | ROUGE-1 | Generation |
| Machine Translation | WMT19 En–Zh, WMT19 Zh–En ( | BLEU | Generation |

where $p_n$ is the clipped (modified) $n$-gram precision of $C$ with respect to the references, and $w_n$ are typically uniform (e.g., $w_n = \frac{1}{N}$ with $N = 4$). The brevity penalty is:

$$\mathrm{BP} = \min\Big(1, \exp\Big(1 - \frac{r}{c}\Big)\Big), \tag{12}$$

with $c$ the length of $C$ and $r$ the length of the closest reference. Higher BLEU indicates closer agreement with the references.

### D.2 ROUGE-1 (UNIGRAM ROUGE)

ROUGE-1 is a recall-oriented metric for summarization and expansion that measures unigram overlap between the candidate and references. It is computed as:

$$\text{ROUGE-1} = \frac{\sum_{S \in \text{Refs}} \sum_{u \in S} \min\big(\mathrm{Count}_{\text{cand}}(u), \mathrm{Count}_S(u)\big)}{\sum_{S \in \text{Refs}} \sum_{u \in S} \mathrm{Count}_S(u)}. \tag{13}$$

where $\mathrm{Count}_{\text{cand}}(u)$ and $\mathrm{Count}_S(u)$ denote unigram counts in the candidate and reference $S$, respectively. Larger ROUGE-1 implies better coverage of reference content.

### D.3 MSE (MEAN SQUARED ERROR)

For STS, we treat similarity prediction as regression and report MSE to measure the discrepancy between predictions and ground-truth labels:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{14}$$

where $y_i$ and $\hat{y}_i$ are the true and predicted similarity scores for the $i$-th example. Lower MSE indicates better performance.

### D.4 ACCURACY

For classification tasks, we use accuracy, defined as the fraction of correctly predicted instances:

$$\mathrm{Accuracy} = \frac{\#\mathrm{Correct}}{\#\mathrm{Total}}. \tag{15}$$

### D.5 PER-INSTANCE SUPERVISION SCORES

For the offline supervision described in Section 2.2, each query–ordering pair is assigned a per-instance task score. For classification, we use the correctness indicator, $s_\ell(x^i_{\mathrm{qu}}, c) = \mathbb{I}[\hat{y}^c_{i,\ell} = y^i_{\mathrm{qu}}]$, which equals 1 for a correct prediction and 0 otherwise. For STS, we use the squared prediction error, $s_\ell(x^i_{\mathrm{qu}}, c) = (\hat{y}^c_{i,\ell} - y^i_{\mathrm{qu}})^2$. Averaging these scores over evaluation queries yields classification accuracy and STS MSE, respectively. Higher scores are preferred for classification, whereas lower scores are preferred for STS.

These per-instance task scores serve as regression targets for the ordering predictor. The MSE training objective in Section 2.3 measures the discrepancy between the predictor's estimated scores and these observed task scores.

## E ORDERING PREDICTOR ARCHITECTURE

In this section, we describe the overall architecture that integrates pre-trained BERT representations with a lightweight projection module and subsequent contextual refinement using a BERT-based backbone.

We focus on the embedding-first predictor, ADOICLE, under the main setting of $k = 4$ demonstrations. The predictor receives four demonstration embeddings and a separately encoded query embedding, each with dimensionality $d = 768$, and produces a scalar estimate of the target LLM's task performance. All tensor shapes below omit the batch dimension.

**Input Representation.** Let $x$ denote the input query and $\mathcal{D}_x = \{(x_j, y_j)\}_{j=1}^4$ denote the retrieved demonstrations. We use a pre-trained `BERT-base-uncased` encoder $f_e$ to obtain

$$\mathbf{v}_q = f_e(x) \in \mathbb{R}^{768}, \qquad \mathbf{h}_j = f_e(g(x_j, y_j)) \in \mathbb{R}^{768},$$

where $g(x_j, y_j)$ serializes the demonstration input and its corresponding label or target output using a fixed template. We use the final-layer hidden state of the `[CLS]` token as the sequence representation.

The representation encoder is frozen and evaluated with dropout disabled during predictor training and inference. Its parameters are separate from those of the trainable ordering predictor. Consequently, the query and demonstration embeddings can be cached and reused across candidate orderings.

Given four contextual embeddings $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \mathbf{h}_4 \in \mathbb{R}^{768}$, we concatenate them to form a single representation. For a candidate permutation $\pi$, the concatenation follows the corresponding demonstration ordering:

$$\mathbf{h}^{\pi}_{\mathrm{concat}} = [\mathbf{h}_{\pi(1)}; \mathbf{h}_{\pi(2)}; \mathbf{h}_{\pi(3)}; \mathbf{h}_{\pi(4)}] \in \mathbb{R}^{3072}.$$

The position of each embedding block explicitly encodes its position in the candidate ordering. Different permutations therefore provide order-dependent inputs to the predictor without requiring the demonstrations to be re-encoded.

**Projection Layer.** To ensure dimensional compatibility with the subsequent BERT layers, we apply a linear transformation that maps the 3072-dimensional concatenated vector to the hidden size of the BERT backbone:

$$\mathbf{z}_\pi = \mathbf{W}_p \mathbf{h}^\pi_{\text{concat}} + \mathbf{b}_p, \qquad \mathbf{W}_p \in \mathbb{R}^{768\times 3072}, \quad \mathbf{b}_p \in \mathbb{R}^{768}.$$

This projection compresses the ordered demonstration features into a single context representation. The projection parameters are shared across queries and candidate orderings.

**Query–Context Fusion.** We construct a two-token input sequence by stacking the query embedding and the projected context embedding:

$$\mathbf{U}_\pi = \text{Stack}\left(\mathbf{v}_q, \mathbf{z}_\pi\right) \in \mathbb{R}^{2\times 768}.$$

Here, Stack places the two vectors in separate rows: the first token represents the query, and the second represents the ordered demonstration context.

We add position embeddings and token-type embeddings to distinguish the two token positions and their roles:

$$\mathbf{H}^{(0)}_\pi = \text{Dropout}\left(\text{LayerNorm}\left(\mathbf{U}_\pi + \mathbf{E}_{\text{pos}} + \mathbf{E}_{\text{type}}\right)\right),$$

where $\mathbf{E}_{\text{pos}}, \mathbf{E}_{\text{type}} \in \mathbb{R}^{2\times 768}$. The query and context tokens use position indices 0 and 1 and token-type indices 0 and 1, respectively. This input-processing stage directly operates on the supplied continuous representations rather than performing an additional word-embedding lookup.

**Backbone Integration.** The resulting two-token sequence is processed by six Transformer encoder layers:

$$\mathbf{H}^{(r)}_\pi = \text{Transformer}_r\left(\mathbf{H}^{(r-1)}_\pi\right), \qquad r = 1, \ldots, 6,$$

with $\mathbf{H}^{(r)}_\pi \in \mathbb{R}^{2\times 768}$. Both token positions are valid and use bidirectional attention.

We initialize these layers from the first six encoder layers of `BERT-base-uncased`. The position embeddings, token-type embeddings, and input LayerNorm are initialized from the same checkpoint. These predictor components are trainable and are separate from the frozen representation encoder.

The attention layers model interactions between the query and the aggregated ordered-demonstration representation. Individual demonstrations are fused through the preceding concatenation and projection rather than represented as separate tokens within these six layers.

**Pooling and Prediction Head.** We apply mean pooling over the two output token representations:

$$\mathbf{u}_\pi = \frac{1}{2}\left(\mathbf{H}^{(6)}_{\pi,1} + \mathbf{H}^{(6)}_{\pi,2}\right) \in \mathbb{R}^{768}.$$

A two-layer prediction head maps the pooled representation to a scalar performance estimate:

$$\mathbf{a}_\pi = \text{GELU}\left(\mathbf{W}_h \mathbf{u}_\pi + \mathbf{b}_h\right),$$

$$\hat{s}_\ell(x, c_\pi) = \mathbf{w}_o^\top \text{Dropout}\left(\mathbf{a}_\pi; p = 0.3\right) + b_o,$$

where

$$\mathbf{W}_h \in \mathbb{R}^{256\times 768}, \qquad \mathbf{b}_h, \mathbf{w}_o \in \mathbb{R}^{256}, \qquad b_o \in \mathbb{R}.$$

The output layer uses a linear activation to predict the task-specific performance score.

The projection layer and prediction head are randomly initialized and trained jointly with the six-layer backbone. For a fixed target LLM $\ell$, the predictor parameters $\theta_\ell$ are shared across all queries and candidate orderings. Training uses the mean squared error objective described in the main text, with supervision generated offline by the corresponding target LLM.

**Rationale.** This architecture separates reusable text encoding from candidate-specific performance prediction. The ordered concatenation preserves the demonstration ordering information, while the projection matches the hidden dimensionality of the predictor backbone. The two-token formulation allows the query to interact with the ordered context before scalar prediction. Embedding reuse reduces repeated encoding across candidate orderings, although each candidate still requires projection and predictor evaluation.

Table 10: The results of the plug-and-play setting. **BM25** and **CD** first retrieve the candidate demonstrations. **With ordering predictor** keeps this retriever fixed and applies ADOICL to the retrieved candidates demonstrations and form the final in-context. These results show that ADOICL can be seamlessly plugged into different ICL pipelines without retriever retraining the ordering.

| | With ordering predictor | | | Without ordering predictor | | |
|---|---|---|---|---|---|---|
| LLMs | GLM | Llama | Qwen | GLM | Llama | Qwen |
| **Classification (Emotion) Accuracy (%) ↑** | | | | | | |
| BM25 | 61.3 | 31.1 | 65.2 | 54.3 | 28.3 | 60.3 |
| CD | 65.1 | 40.4 | 60.6 | 55.8 | 37.0 | 53.8 |
| **Summarization (Gigaword) ROUGE-1 ↑** | | | | | | |
| BM25 | 0.031 | 0.023 | 0.025 | 0.028 | 0.020 | 0.023 |
| CD | 0.037 | 0.064 | 0.155 | 0.032 | 0.056 | 0.151 |

Table 11: Experimental results on additional LLMs. The best results are highlighted in **bold**, and the second-best results are underlined.

| LLMs | Llama3.2-1b | Llama-3.3-70b | DeepSeek-32b | Qwen3.5 4b | Qwen3.5 9b |
|---|---|---|---|---|---|
| | Emotion-Accuracy (%) ↑ | | | | |
| BM25 | 60.7 | 68.4 | 70.2 | 69.1 | 69.1 |
| CR | 40.5 | 57.9 | 60.1 | 55.2 | 56.0 |
| CD | 40.6 | 58.7 | 61.0 | 54.1 | 57.4 |
| Kate | 64.4 | 69.2 | 71.6 | 67.0 | 72.4 |
| DKNN | 46.7 | 58.9 | 59.3 | 69.4 | 54.0 |
| ICCL | 64.1 | 69.0 | 70.5 | 67.8 | 71.6 |
| GlobalE | 39.1 | 59.8 | 57.6 | 52.9 | 51.2 |
| ADOICLE | **66.9** | **77.5** | **79.0** | **73.6** | **75.9** |
| | SST5-Accuracy (%) ↑ | | | | |
| BM25 | 36.1 | 58.2 | 55.6 | 52.6 | 54.1 |
| CR | 29.6 | 54.7 | 51.2 | 48.9 | 50.8 |
| CD | 32.4 | 45.3 | 48.1 | 46.2 | 47.0 |
| Kate | 42.1 | 56.8 | 55.4 | 53.1 | 49.9 |
| DKNN | 38.0 | 54.2 | 56.9 | 54.1 | 51.7 |
| TTF | 36.7 | 59.3 | 58.4 | 55.8 | 54.3 |
| ICCL | 38.6 | 57.0 | 55.1 | 52.8 | 52.3 |
| GlobalE | 36.7 | 58.9 | 57.0 | 55.3 | 53.6 |
| ADOICLE | **46.5** | **65.4** | **63.7** | **60.2** | **58.6** |
| | STS14-MSE ↓ | | | | |
| BM25 | 5.92 | 1.47 | 1.34 | 1.95 | 1.71 |
| CR | 6.08 | 1.31 | 1.22 | 1.82 | 1.19 |
| CD | 6.03 | 1.10 | 1.14 | 1.79 | 1.08 |
| Kate | 6.01 | 1.09 | 1.33 | 1.46 | 1.78 |
| DKNN | - | - | - | - | - |
| TTF | - | - | - | - | - |
| ICCL | 5.88 | 1.08 | 1.20 | 1.60 | 1.04 |
| GlobalE | 5.72 | 1.24 | 1.27 | 1.55 | 1.23 |
| ADOICLE | **5.33** | **0.90** | **0.88** | 1.46 | **0.98** |

## F Cross-Demonstration-Number Generalization

We further evaluate whether ADOICL can generalize across different demonstration numbers. This experiment is designed to examine whether the ordering predictor is tied to a single demonstration budget. Specifically, we fix the input arity of the ordering predictor to 4 and evaluate ADOICL under $k = 3$, $k = 4$, and $k = 5$ demonstrations. As shown in Table 15, ADOICL remains effective across different demonstration numbers on both classification and generation tasks. On Emotion, the accuracy increases from 69.2% at $k = 3$ to 73.3% at $k = 5$. On Gigaword, ROUGE-1 also improves from 0.064 to 0.069. These results indicate that ADOICL is not restricted to a single fixed demonstration number and can remain effective when the demonstration budget changes.

Table 12: The results 100 shots. At this stage, we use the `BigBird-RoBERTa-large` pretrained model for vectorization.

| Test Data | Gigaword | | | Emotion | | | SST5 | | |
|---|---|---|---|---|---|---|---|---|---|
| LLMs | GLM4 9B | Llama3.1 8B | Qwen2.5 7B | GLM4 9B | Llama3.1 8B | Qwen2.5 7B | GLM4 9B | Llama3.1 8B | Qwen2.5 7B |
| Shot | 100 | | | 100 | | | 100 | | |
| Metric | ROUGE-1 ↑ | | | Accuracy (%) ↑ | | | Accuracy (%) ↑ | | |
| BM25 | 0.35 | 0.37 | 0.36 | 72.5 | 68.7 | 68.3 | 52.9 | 48.2 | 50.6 |
| CR | 0.35 | 0.37 | 0.34 | 63.6 | 56.4 | 56.0 | 46.9 | 43.4 | 36.2 |
| CD | 0.23 | 0.37 | 0.34 | 65.5 | 65.8 | 63.1 | 52.3 | 51.1 | 49.5 |
| ICCL | 0.38 | 0.48 | 0.35 | 71.9 | 69.2 | 63.4 | 53.8 | 48.4 | 20.0 |
| Kate | 0.38 | 0.37 | 0.35 | 72.0 | 69.4 | 63.0 | 54.3 | 50.2 | 21.7 |
| DKNN | 0.35 | 0.37 | 0.36 | 35.9 | 57.5 | 57.0 | 58.4 | 52.7 | 43.9 |
| TTF | 0.39 | 0.51 | 0.43 | 63.4 | 58.1 | 58.0 | 53.7 | 51.2 | 43.1 |
| ADOICLE | **0.44** | **0.54** | **0.48** | **78.1** | **73.0** | **70.5** | **59.4** | **60.3** | **55.9** |

Table 13: Effect of different training-set construction strategies. All strategies follow the same query/in-context split ratio and avoid overlap between query instances and in-context demonstrations.

| LLM | Strategy | SST5 Acc. ↑ | Gigatiny R-1 ↑ | WMT19 En-Zh BLEU ↑ | STS14 MSE ↓ |
|---|---|---|---|---|---|
| GLM 4-9B | Random split | 51.6 | 0.235 | 0.232 | 0.768 |
| GLM 4-9B | Stratified split | 50.2 | 0.237 | 0.233 | 0.772 |
| GLM 4-9B | Cluster-based split | 52.0 | 0.233 | 0.236 | 0.764 |
| LLaMA 3.1-8B | Random split | 39.9 | 0.133 | 0.026 | 5.871 |
| LLaMA 3.1-8B | Stratified split | 40.3 | 0.131 | 0.024 | 5.826 |
| LLaMA 3.1-8B | Cluster-based split | 39.6 | 0.135 | 0.026 | 5.714 |
| Qwen2.5-7B | Random split | 52.3 | 0.341 | 0.136 | 0.645 |
| Qwen2.5-7B | Stratified split | 52.6 | 0.322 | 0.133 | 0.649 |
| Qwen2.5-7B | Cluster-based split | 53.1 | 0.338 | 0.138 | 0.639 |

## G Details of ADOICL Variants

**(1) ADOICL for Vision Tasks.** We extend ADOICL to vision tasks by using CLIP as the encoder. For each visual demonstration, we use $\boldsymbol{v}^{ij}_{\text{demo}} = \text{Concat}(f_{\text{img}}(x^{ij}_{\text{re}}), f_{\text{text}}(y^{ij}_{\text{re}}))$ as its demonstration embedding. Different demonstration orderings are represented by concatenating the corresponding demonstration embeddings, and the ordering predictor is used to score candidate orders. All other components remain unchanged.

**(2) Few-Shot Training-Data ADOICL.** In few-shot training-data settings, the number of query instances available for constructing offline supervision can be limited. To increase the diversity of candidate orderings, we use multiple retrieval methods, such as BM25, CD, and CR, to generate additional candidate demonstration sets. Although the number of training queries remains small, each query can induce multiple query–ordering pairs, providing additional supervision for training the ordering predictor.

**(3) Many-Shot Clustering-Based Group-Level ADOICL.** For many-shot settings with a large demonstration budget $k$, exhaustive instance-level ordering over all $k$ demonstrations is computationally infeasible. We therefore use a clustering-based group-level scalability variant. Given the retrieved demonstrations $\mathcal{N}_k(x) = \{(x^1_{\text{in}}, y^1_{\text{in}}), \ldots, (x^k_{\text{in}}, y^k_{\text{in}})\}$, we first compute their embeddings $\boldsymbol{v}^j_{\text{demo}} = f_{\text{e}}(g(x^j_{\text{in}}, y^j_{\text{in}}))$ for $j = 1, \ldots, k$, and cluster them into $G$ semantically coherent groups $\{\mathcal{G}_1, \ldots, \mathcal{G}_G\}$. Each group $\mathcal{G}_g$ contains a subset of the retrieved demonstrations and is represented by a pooled group embedding $\boldsymbol{v}_{\mathcal{G}_g} = \text{Pool}(\{\boldsymbol{v}^j_{\text{demo}} \mid (x^j_{\text{in}}, y^j_{\text{in}}) \in \mathcal{G}_g\})$, where $\text{Pool}(\cdot)$ can be implemented by mean pooling. ADOICL then performs ordering over the $G$ groups rather than over all $k$ individual demonstrations. Within each group, demonstrations are arranged according to query–demonstration similarity. The final in-context sequence is obtained by concatenating the ordered groups and their within-group demonstrations. This variant is intended as a practical scalability mechanism for many-shot settings, rather than an exhaustive optimization over all $k!$ instance-level permutations.

Table 14: Effect of the number of sampled candidate orders $M$ for offline supervision construction. At test time, the main $k = 4$ setting scores the full $4!$ candidate pool with the ordering predictor.

| LLM | $M$ | SST5 Acc. ↑ | Gigatiny R-1 ↑ | WMT19 En-Zh BLEU ↑ |
|---|---|---|---|---|
| GLM 4-9B | 1 | 49.4 | 0.221 | 0.213 |
| GLM 4-9B | 3 | 52.1 | 0.232 | 0.226 |
| GLM 4-9B | 5 | 51.6 | 0.235 | 0.232 |
| GLM 4-9B | 10 | 53.4 | 0.234 | 0.229 |
| GLM 4-9B | 24 | 54.7 | 0.238 | 0.227 |
| LLaMA 3.1-8B | 1 | 37.1 | 0.123 | 0.024 |
| LLaMA 3.1-8B | 3 | 38.4 | 0.126 | 0.025 |
| LLaMA 3.1-8B | 5 | 39.9 | 0.133 | 0.026 |
| LLaMA 3.1-8B | 10 | 43.0 | 0.136 | 0.026 |
| LLaMA 3.1-8B | 24 | 43.7 | 0.137 | 0.026 |

Table 15: Cross-demonstration-number generalization results. The input arity of the ordering predictor is fixed to 4, while the demonstration number $k$ varies.

| Metric | Dataset | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|---|
| Accuracy (%) ↑ | Emotion | 69.2 | 70.8 | 73.3 |
| ROUGE-1 ↑ | Gigaword | 0.064 | 0.065 | 0.069 |

## H Ablation Results for ADOICLS

Table 17 reports the ablation results of ADOICLS. Overall, the results show that the three core components remain important for ADOICLS: similarity-based retrieval, the ordering predictor, and the dual-input architecture.

(I) **Similarity-based retrieval.** Similarity-based retrieval performs better overall than the diversity-based retrieval strategy. It improves accuracy by 6.58% on average and reduces MSE by 0.110 across target LLMs. Thus, we also use similarity-based retrieval as the default retrieval strategy for ADOICLS.

(II) **Ordering predictor.** Removing the ordering predictor consistently degrades performance, causing an average accuracy drop of 9.08% and an MSE increase of 0.081. This confirms that explicitly predicting ordering performance is crucial for ADOICLS to select high-performing demonstration orderings.

(III) **Dual-input architecture.** The dual-input architecture consistently outperforms the single-input variant. It improves accuracy by 5.03% on average, reduces MSE by 0.149, and yields an average gain of 0.023 on generation metrics. These results show that separately modeling the query and contextual demonstrations remains important for ordering-performance prediction in ADOICLS.

## I Generalization Results for ADOICLS

We further report the generalization results of ADOICLS. Overall, ADOICLS also shows stable transferability across target LLMs, datasets, and tasks, indicating that the ordering predictor can capture transferable ordering-performance patterns beyond the training setting.

**(1) Cross-LLM generalization.** As shown in Table 18, the ADOICLS ordering predictor trained on one target LLM can be transferred to another without retraining. Although cross-LLM transfer is slightly weaker than training on the target LLM itself, it remains effective. For example, when tested on Llama 3.1-8B, the predictor trained on GLM 4-9B achieves 52.3% accuracy on Emotion and 0.063 ROUGE-1 on Gigaword, while the predictor trained on Qwen2.5-7B achieves 48.6% accuracy and 0.066 ROUGE-1. These results suggest that ADOICLS can generalize across target LLMs.

**(2) Cross-dataset generalization.** Table 19 evaluates whether the ADOICLS ordering predictor transfers across datasets. When trained on SST5 and tested on Emotion, ADOICLS achieves 63.4%,

Table 16: Effect of the number of demonstrations $k$.

| LLM | $k$ | SST5 Acc. ↑ | Gigatiny R-1 ↑ | WMT19 En-Zh BLEU ↑ |
|---|---|---|---|---|
| GLM 4-9B | 2 | 47.8 | 0.213 | 0.207 |
| GLM 4-9B | 3 | 50.8 | 0.230 | 0.223 |
| GLM 4-9B | 4 | 51.6 | 0.235 | 0.232 |
| GLM 4-9B | 5 | 55.3 | 0.267 | 0.262 |
| LLaMA 3.1-8B | 2 | 35.9 | 0.130 | 0.024 |
| LLaMA 3.1-8B | 3 | 37.2 | 0.133 | 0.026 |
| LLaMA 3.1-8B | 4 | 39.9 | 0.133 | 0.026 |
| LLaMA 3.1-8B | 5 | 42.7 | 0.144 | 0.029 |

Table 17: Ablation results for ADOICLS.

| Ablation setting | | | Results | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Similarity-based retrieval | Ordering predictor | Dual-input architecture | Accuracy (%) ↑ Classification: Emotion | | | Accuracy (%) ↑ Classification: SST5 | | | ROUGE-1 ↑ Summarization: Gigaword | | | ROUGE-1 ↑ Expansion: Gigatiny | | |
| | | | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ✓ | ✓ | ✓ | **71.4** | **58.1** | **68.6** | **52.0** | **40.3** | **53.8** | **0.069** | **0.078** | **0.183** | **0.239** | **0.136** | **0.347** |
| × | ✓ | ✓ | 60.9 | 45.6 | 58.4 | 50.9 | 38.1 | 50.8 | 0.067 | 0.067 | 0.176 | 0.211 | 0.130 | 0.329 |
| ✓ | × | ✓ | 56.1 | 44.8 | 59.3 | 46.2 | 34.4 | 48.9 | 0.056 | 0.050 | 0.120 | 0.197 | 0.129 | 0.297 |
| ✓ | ✓ | × | 65.2 | 48.7 | 66.8 | 48.9 | 37.0 | 47.4 | 0.061 | 0.063 | 0.158 | 0.218 | 0.125 | 0.282 |
| Ablation setting | | | MSE ↓ STS: STS14 | | | MSE ↓ STS: STSB | | | BLEU ↑ Translation: WMT19 En-Zh | | | BLEU ↑ Translation: WMT19 Zh-En | | |
| Similarity-based retrieval | Ordering predictor | Dual-input architecture | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ✓ | ✓ | ✓ | **0.752** | **5.830** | 0.671 | **0.739** | **1.510** | **0.641** | 0.231 | **0.029** | **0.139** | **0.512** | **0.098** | **0.582** |
| × | ✓ | ✓ | 0.792 | 5.934 | 0.721 | 0.771 | 1.821 | 0.762 | **0.236** | 0.024 | 0.133 | 0.472 | 0.086 | 0.552 |
| ✓ | × | ✓ | 0.797 | 6.038 | 0.685 | 0.787 | 1.638 | 0.685 | 0.201 | 0.021 | 0.130 | 0.466 | 0.091 | 0.514 |
| ✓ | ✓ | × | 0.781 | 6.318 | **0.667** | 0.779 | 1.753 | 0.736 | 0.220 | 0.023 | 0.133 | 0.468 | 0.093 | 0.519 |

50.7%, and 60.5% accuracy on GLM, Llama, and Qwen, respectively, with an average accuracy of 58.2%. When trained on STSB and tested on STS14, it obtains 0.9, 5.9, and 0.8 MSE across the three target LLMs. These results show that the learned ordering signals remain effective under different datasets.

**(3) Cross-task generalization.** We further evaluate cross-task transfer by training the ADOICLS ordering predictor on Emotion for classification and testing it on Gigaword for summarization. As shown in Table 18, ADOICLS achieves 0.058, 0.063, and 0.164 ROUGE-1 on GLM, Llama, and Qwen, respectively, with an average ROUGE-1 of 0.095. Although classification and summarization have different output formats, these results indicate that the predictor can still transfer to a generation task and capture task-robust ordering patterns.

## J Performance across Repeated Runs

**Evaluation.** Table 23 reports SST5 accuracy as mean ± standard deviation over three runs for GLM, Llama, and Qwen. These results characterize both average performance and run-to-run variability.

**Mean performance.** ADOICLE achieves accuracies of $51.6 \pm 2.7\%$, $39.9 \pm 3.0\%$, and $52.3 \pm 2.6\%$ on GLM, Llama, and Qwen, respectively. ADOICLS obtains the highest mean accuracy on all three models, reaching $52.0 \pm 2.5\%$, $40.3 \pm 2.8\%$, and $53.8 \pm 3.2\%$, respectively.

The strongest baselines by mean accuracy are ICCL on GLM ($48.4 \pm 2.0\%$), TTF on Llama ($37.8 \pm 2.4\%$), and BM25 on Qwen ($50.0 \pm 3.1\%$). Relative to these baselines, ADOICLE improves mean accuracy by 3.2, 2.1, and 2.3 percentage points, while ADOICLS improves it by 3.6, 2.5, and 3.8 percentage points, respectively. Thus, both variants retain an advantage in mean performance across the three evaluated LLMs.

Table 18: The results of cross-LLM generalization for ADOICLS. **Training LLM**: the LLM used to provide performance scores for constructing ADOICLS training data. **Testing LLM**: the LLM used to construct the in-context during inference.

| Dataset & Metric | Emotion & Accuracy (%) ↑ | | | Gigaword & ROUGE-1 ↑ | | |
|---|---|---|---|---|---|---|
| Testing LLM | Llama | | | Llama | | |
| Training LLM | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ADOICLS | 52.3 | 58.1 | 48.6 | 0.063 | 0.078 | 0.066 |

Table 19: The results of cross-dataset generalization for ADOICLS. **Training dataset**: the dataset used to train the ADOICLS ordering predictor. **Test dataset**: the dataset on which the ordering predictor is applied to construct the final in-context prompt.

| LLMs | GLM | Llama | Qwen | GLM | Llama | Qwen |
|---|---|---|---|---|---|---|
| Metric | Accuracy (%) ↑ | | | MSE ↓ | | |
| Training Dataset | SST5 | | | STSB | | |
| Testing Dataset | Emotion | | | STS14 | | |
| ADOICLS | 63.4 | 50.7 | 60.5 | 0.9 | 5.9 | 0.8 |

Table 20: Results on CIFAR-10 dataset.

| Metric | Accuracy (%) ↑ | | |
|---|---|---|---|
| LLMs | 4B | 8B | 26B |
| CR | 90.0 | 92.1 | 96.7 |
| CD | 86.1 | 91.3 | 97.1 |
| Kate | 92.4 | 92.1 | 98.0 |
| ICL-rerank | 93.1 | 91.2 | 97.7 |
| ADOICL | 96.3 | 94.8 | 99.3 |

Table 21: Results with few-shot (30-query) training data for ADOICLE.

| Dataset & Metric | Emotion & Accuracy (%) ↑ | | |
|---|---|---|---|
| Shots | GLM | Llama | Qwen |
| 30 | 62.8 | 49.7 | 61.2 |
| Dataset & Metric | Gigaword & ROUGE-1 ↑ | | |
| 30 | 0.059 | 0.075 | 0.162 |

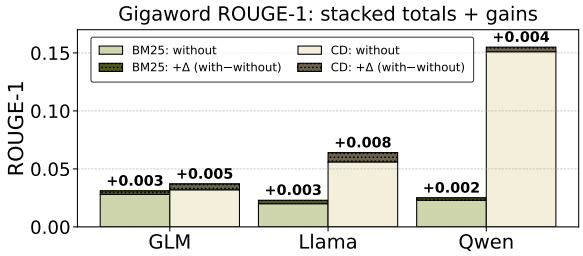


Figure 6: Results on Gigaword under the plug-and-play setting.

**Run-to-run variability.** The standard deviations range from 2.6 to 3.0 percentage points for ADOICLE and from 2.5 to 3.2 for ADOICLS, within the overall range observed among the baselines (2.0–3.5). The improvements in mean accuracy therefore occur with broadly comparable run-to-run variability, rather than uniformly lower variance. These three-run summaries support the mean-performance advantage on SST5, but do not establish statistical significance or imply that ADOICL wins in every run.

## K LIMITATIONS

ADOICL requires labeled data and offline LLM evaluations to construct performance supervision, together with additional computation for predictor training. These upfront costs can be amortized over repeated use, but may limit its practical benefits when supervision is expensive or the deployment workload is small. The quality of the selected ordering depends on both predictor accuracy and candidate-pool coverage. Although ADOICLE reduces repeated text encoding through embedding reuse, exhaustive evaluation still requires scoring $k!$ candidate orderings. For many-shot settings, the grouping-based variant restricts the search to group-level permutations and uses similarity-based ordering within groups, which may exclude better instance-level orderings. Finally, the main ADOICLE implementation uses a fixed input arity, so changes in the demonstration number require additional input adaptation. While our experiments provide evidence of transfer across LLMs, datasets, tasks, and demonstration numbers, these results are limited to the evaluated set-

Table 22: Ablation results for ADOICLE. The ablation results of ADOICLS are in Section H (In the Appendix)

| Ablation setting | | | Results | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Similarity-based retrieval | Ordering predictor | Dual-input architecture | MSE ↓ STS: STS14 | | | MSE ↓ STS: STSB | | | BLEU ↑ Translation: WMT19 En-Zh | | | BLEU ↑ Translation: WMT19 Zh-En | | |
| | | | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| ✓ | ✓ | ✓ | **0.768** | **5.871** | **0.645** | **0.752** | **1.486** | **0.654** | 0.232 | **0.026** | **0.136** | **0.495** | **0.095** | **0.565** |
| × | ✓ | ✓ | 0.784 | 5.960 | 0.734 | 0.760 | 1.863 | 0.748 | **0.235** | 0.025 | 0.132 | 0.480 | 0.083 | 0.547 |
| ✓ | × | ✓ | 0.797 | 6.038 | 0.685 | 0.787 | 1.638 | 0.685 | 0.201 | 0.021 | 0.130 | 0.466 | 0.091 | 0.514 |
| ✓ | ✓ | × | 0.783 | 6.239 | 0.673 | 0.770 | 1.784 | 0.730 | 0.222 | 0.022 | 0.132 | 0.464 | 0.092 | 0.526 |

Table 23: SST5 accuracy (%, mean ± standard deviation over three runs) across three LLMs.

| Method | GLM | Llama | Qwen |
|---|---|---|---|
| BM25 | $34.2 \pm 3.2$ | $29.4 \pm 2.9$ | $50.0 \pm 3.1$ |
| CR | $48.3 \pm 3.5$ | $34.4 \pm 2.6$ | $46.2 \pm 3.0$ |
| CD | $45.0 \pm 2.7$ | $34.2 \pm 3.4$ | $45.7 \pm 2.5$ |
| Kate | $46.2 \pm 2.9$ | $34.4 \pm 2.1$ | $48.9 \pm 3.3$ |
| PLR$^{\dagger,*}$ | $45.7 \pm 2.4$ | $33.9 \pm 3.2$ | $48.2 \pm 2.8$ |
| CDS$^{*}$ | $45.4 \pm 3.1$ | $33.7 \pm 2.7$ | $47.8 \pm 2.3$ |
| DKNN$^{\dagger}$ | $48.0 \pm 2.2$ | $36.7 \pm 2.9$ | $44.1 \pm 3.4$ |
| TTF$^{\dagger}$ | $47.5 \pm 2.8$ | $37.8 \pm 2.4$ | $46.2 \pm 3.1$ |
| ICCL$^{*}$ | $48.4 \pm 2.0$ | $35.5 \pm 2.7$ | $42.1 \pm 2.9$ |
| GlobalE$^{*}$ | $39.6 \pm 3.4$ | $33.7 \pm 2.6$ | $45.0 \pm 3.5$ |
| ADOICLE$^{\dagger,*}$ | $\underline{51.6} \pm 2.7$ | $\underline{39.9} \pm 3.0$ | $\underline{52.3} \pm 2.6$ |
| ADOICLS$^{\dagger,*}$ | $\mathbf{52.0} \pm 2.5$ | $\mathbf{40.3} \pm 2.8$ | $\mathbf{53.8} \pm 3.2$ |

tings. Performance under larger distribution shifts and substantially different demonstration budgets requires further investigation and may benefit from additional supervision or predictor adaptation.

## L Ordering Quality with Fixed Demonstration Sets

**Evaluation protocol.** We evaluate demonstration ordering while holding the retrieved demonstration set fixed for each test query. Under the main setting of $k = 4$, we enumerate all $4! = 24$ permutations of the same four demonstrations. The query, demonstration contents, prompt template, and target LLM configuration remain unchanged across candidate orderings. This protocol isolates the effect of ordering from differences in demonstration retrieval or selection. We evaluate GLM, Llama, and Qwen on Emotion, SST5, Gigaword, and Gigatiny, using accuracy for classification and ROUGE-1 for generation.

**Compared ordering strategies.** We compare six ordering strategies and an oracle upper bound. *Original* preserves the order returned by the retriever, whereas *Reverse* reverses that order. *Random* reports the expected performance of uniformly sampling a permutation, computed by averaging the observed scores over all 24 permutations for each query. *Best fixed ordering* selects a single permutation of retrieval-rank positions on validation data and applies it uniformly to all test queries within the corresponding model–dataset setting. It therefore provides a learned but query-independent ordering strategy. ADOICLE and ADOICLS instead score the candidate permutations for each query and select the one with the highest predicted performance. Finally, *Oracle* selects the permutation with the highest observed task score for each query. Oracle uses ground-truth outcomes only for evaluation and is not an inference-time method.

**Oracle gap.** Let $\Pi_4$ denote the set of all permutations of four demonstrations, and let $s_i(\pi)$ denote the observed task score for query $i$ under permutation $\pi$. For an ordering method $m$ that selects $\hat{\pi}_i^{(m)}$, we define its average oracle gap as

$$\mathrm{Gap}(m) = \frac{1}{N} \sum_{i=1}^{N} \left[ \max_{\pi \in \Pi_4} s_i(\pi) - s_i\left(\hat{\pi}_i^{(m)}\right) \right],$$

Table 24: Ordering performance with fixed demonstration sets. Each dataset is evaluated using GLM, Llama, and Qwen. Classification results are reported as accuracy (%); generation results are reported as ROUGE-1 on a 0–1 scale. Best fixed ordering uses a single permutation selected on validation data and applied to all test queries. Oracle selects the best permutation for each query using ground-truth outcomes and is included only as an evaluation upper bound. Bold indicates the best non-oracle result in each column.

| Task | Classification | | | Classification | | | Summarization | | | Expansion | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | Emotion | | | SST5 | | | Gigaword | | | Gigatiny | | |
| Metric | Accuracy (%) ↑ | | | Accuracy (%) ↑ | | | ROUGE-1 ↑ | | | ROUGE-1 ↑ | | |
| LLM | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| Original | 58.5 | 42.0 | 61.9 | 43.8 | 35.5 | 47.4 | 0.056 | 0.047 | 0.123 | 0.183 | 0.135 | 0.274 |
| Reverse | 53.8 | 45.4 | 57.7 | 47.0 | 33.0 | 49.7 | 0.051 | 0.052 | 0.111 | 0.198 | 0.122 | 0.307 |
| Random | 58.9 | 42.2 | 62.4 | 44.3 | 33.1 | 47.0 | 0.056 | 0.047 | 0.124 | 0.181 | 0.134 | 0.277 |
| Best fixed ordering | 56.4 | 47.4 | 58.8 | 48.9 | 34.1 | 49.9 | 0.053 | 0.052 | 0.114 | 0.206 | 0.126 | 0.309 |
| ADOICLE | 70.8 | 57.5 | **69.0** | 51.6 | 39.9 | 52.3 | 0.065 | **0.081** | 0.179 | 0.235 | 0.133 | 0.341 |
| ADOICLS | **71.4** | **58.1** | 68.6 | **52.0** | **40.3** | **53.8** | **0.069** | 0.078 | **0.183** | **0.239** | **0.136** | **0.347** |
| Oracle | 75.6 | 62.7 | 76.1 | 58.2 | 46.4 | 61.2 | 0.078 | 0.087 | 0.199 | 0.261 | 0.163 | 0.394 |

Table 25: Results of cross-dataset generalization. **Training dataset**: the dataset used to train the ADOICL ordering predictor. **Test dataset**: the dataset on which the ordering predictor is applied to construct the final in-context prompt.

| LLMs | GLM | Llama | Qwen | GLM | Llama | Qwen |
|---|---|---|---|---|---|---|
| Metric | Accuracy (%) ↑ | | | MSE ↓ | | |
| Training dataset | SST5 | | | STSB | | |
| Test dataset | Emotion | | | STS14 | | |
| ADOICL | 64.1 | 49.8 | 61.9 | 0.9 | 5.8 | 0.8 |

where $N$ is the number of evaluated queries. For classification, $s_i(\pi)$ is the correctness indicator; for generation, it is the per-example ROUGE-1 score. With these example-averaged metrics, the oracle gap equals the difference between the Oracle result and the method's result in Table 24. Classification gaps are reported in percentage points, whereas generation gaps use the same 0–1 scale as ROUGE-1. Smaller gaps indicate better selection from the candidate pool.

**Comparison with ordering baselines.** Table 24 shows that ADOICLS exceeds all four ordering baselines in all 12 model–dataset settings. ADOICLE achieves the same result in 11 settings. The exception is Gigatiny with Llama, where ADOICLE obtains 0.133 ROUGE-1, slightly below Original at 0.135 and Random at 0.134.

The improvements are particularly pronounced on Emotion. Relative to Original, ADOICLS improves accuracy by 12.9, 16.1, and 6.7 percentage points for GLM, Llama, and Qwen, respectively. For generation, ADOICLS improves Gigaword ROUGE-1 from 0.123 to 0.183 with Qwen and Gigatiny ROUGE-1 from 0.274 to 0.347 with the same model. These comparisons demonstrate that selecting an ordering can improve downstream performance without changing the retrieved demonstration set.

**Query-dependent versus fixed ordering.** Both ADOICLE and ADOICLS outperform Best fixed ordering in all 12 settings. For example, on Emotion with Llama, Best fixed ordering achieves 47.4% accuracy, compared with 57.5% for ADOICLE and 58.1% for ADOICLS. On Gigaword with Qwen, the corresponding ROUGE-1 scores are 0.114, 0.179, and 0.183. These results support choosing an ordering for each query rather than applying a single validation-selected permutation to every test query. Best fixed ordering does not always outperform Original on the test set, indicating that a globally selected permutation need not generalize uniformly to unseen queries.

**Remaining gap to the oracle.** Despite these improvements, both predictors remain below Oracle in every setting. On Emotion with Llama, ADOICLS reduces the oracle gap from 20.7 percentage points for Original to 4.6 percentage points. On Gigaword with Qwen, it reduces the gap from 0.076

Table 26: Prompt templates for textual tasks.

| Dataset | Instruction |
|---|---|
| Emotion | You are a helpful assistant. Predict the emotion label of the input text. The possible labels are Anger, Fear, Joy, Love, Sadness, and Surprise. Output only the final label. |
| SST5 | You are a helpful assistant. Predict the sentiment label of the input text. The possible labels are very negative, negative, neutral, positive, and very positive. Output only the final label. |
| STS14 / STS-B | You are a helpful assistant. Predict the semantic textual similarity of the input sentence pair. Output only one numerical score between 0 and 5, where a larger value indicates greater similarity. |
| Gigaword | You are a helpful assistant. Summarize the following text and output only the generated summary. |
| WMT19 En–Zh | You are a helpful assistant. Translate the following English text into Chinese and output only the translation. |

Table 27: Results on generation tasks, reported as mean ± standard deviation over three runs. The highest mean in each column is highlighted in **bold**.

| LLMs | GLM | Llama | Qwen | GLM | Llama | Qwen |
|---|---|---|---|---|---|---|
| Metric | ROUGE-1 ↑ | | | BLEU ↑ | | |
| Task & Data | Expansion: Gigatiny | | | Translation: WMT19 Zh-En | | |
| Method | | | | | | |
| BM25 | 0.116±0.002 | 0.116±0.005 | 0.122±0.004 | 0.326±0.015 | 0.078±0.003 | 0.038±0.002 |
| CR | 0.126±0.004 | 0.102±0.007 | 0.224±0.010 | 0.207±0.007 | 0.084±0.003 | 0.066±0.004 |
| CD | 0.116±0.007 | 0.108±0.003 | 0.302±0.017 | 0.424±0.032 | 0.078±0.005 | 0.452±0.014 |
| Kate | 0.205±0.007 | 0.124±0.004 | 0.286±0.010 | 0.460±0.027 | 0.090±0.005 | 0.519±0.026 |
| PLR†,* | 0.191±0.010 | 0.125±0.006 | 0.280±0.008 | 0.449±0.028 | 0.086±0.003 | 0.477±0.021 |
| CDS* | 0.180±0.007 | 0.119±0.004 | 0.284±0.018 | 0.430±0.014 | 0.087±0.007 | 0.469±0.021 |
| DKNN† | 0.192±0.008 | 0.119±0.003 | 0.256±0.013 | 0.487±0.017 | 0.073±0.005 | 0.510±0.016 |
| TTF† | 0.206±0.006 | 0.098±0.002 | 0.283±0.019 | 0.444±0.018 | 0.094±0.003 | 0.503±0.014 |
| ICCL* | 0.203±0.011 | 0.011±0.001 | 0.233±0.010 | 0.304±0.019 | 0.091±0.004 | 0.277±0.013 |
| GlobalE* | 0.142±0.004 | 0.118±0.006 | 0.205±0.013 | 0.373±0.019 | 0.080±0.003 | 0.137±0.004 |
| ADOICLE†,* | **0.233±0.014** | 0.138±0.005 | 0.341±0.027 | **0.515±0.018** | 0.092±0.003 | **0.590±0.022** |
| ADOICLS†,* | 0.229±0.009 | **0.140±0.004** | **0.360±0.013** | 0.492±0.020 | **0.102±0.003** | 0.581±0.041 |

to 0.016 ROUGE-1. However, a gap of 7.4 percentage points remains on SST5 with Qwen, and a gap of 0.047 ROUGE-1 remains on Gigatiny with Qwen. Thus, the predictors recover part of the performance available through ordering selection, while leaving additional room for improvement.

These results evaluate the quality of the selected permutation and its downstream utility. They do not directly measure the accuracy of the complete predicted ranking over all candidate permutations, which requires a separate within-query ranking analysis.

## M Comparison with Other Demonstration-Ordering Methods

We further compare ADOICL with representative demonstration-ordering methods, including DEmO and OptiSeq. These approaches aim to improve ICL performance by learning or optimizing effective demonstration arrangements. However, their ordering objectives are not explicitly optimized toward the downstream utility of candidate orderings under the target LLM.

As shown in Tables 27 and 28, both ADOICL variants consistently outperform existing ordering methods across classification and generation tasks. Specifically, ADOICLS achieves the best results in most settings, while ADOICLE also obtains substantial improvements over DEmO and OptiSeq. These results demonstrate that modeling the relationship among the query, retrieved demonstrations,

Table 28: Comparison with demonstration-ordering methods. Best results are in **bold**.

| Method | Emotion | | | SST5 | | | Gigaword | | | Gigatiny | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen | GLM | Llama | Qwen |
| DEmO* | 50.6 | 35.8 | 51.4 | 44.9 | 33.9 | 44.4 | 0.032 | 0.041 | 0.099 | 0.158 | 0.061 | 0.229 |
| OptiSeq* | 50.3 | 43.6 | 49.9 | 43.6 | 33.8 | 46.3 | 0.040 | 0.039 | 0.127 | 0.157 | 0.117 | 0.237 |
| ADOICLE†, ∗ | 70.8 | 57.5 | **69.0** | 51.6 | 39.9 | 52.3 | 0.065 | **0.081** | 0.179 | 0.235 | 0.133 | 0.341 |
| ADOICLS†, ∗ | **71.4** | **58.1** | 68.6 | **52.0** | **40.3** | **53.8** | **0.069** | 0.078 | **0.183** | **0.239** | **0.136** | **0.347** |

Table 29: Top-1 oracle-order hit rates (%) on Gigatiny. Each method is evaluated against the optimal orderings of its own retrieved demonstration set. Higher is better. The best and second-best results for each LLM are shown in **bold** and underlined, respectively.

| Dataset | LLM | BM25 | CR | CD | Kate | PLR | CDS | DKNN | TTF | ICCL | GlobalE | ADOICLE | ADOICLS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gigatiny | GLM | 4.7 | 5.3 | 3.5 | 3.8 | 6.5 | 5.2 | 4.9 | 4.9 | 4.5 | 6.2 | <u>26.8</u> | **28.0** |
| | Llama | 6.1 | 4.3 | 4.2 | 5.3 | 4.8 | 6.2 | 4.4 | 4.7 | 5.2 | 5.8 | <u>27.5</u> | **27.8** |
| | Qwen | 4.8 | 5.2 | 5.4 | 4.1 | 7.0 | 4.4 | 4.9 | 4.2 | 4.2 | 5.5 | <u>23.6</u> | **24.9** |

candidate orderings, and target-LLM performance provides a more effective criterion for demonstration ordering. Rather than relying on proxy objectives, ADOICL predicts downstream utility to guide the selection of effective demonstration orderings.

## N Ordering Selection Quality on Gigatiny

We evaluate ordering selection quality on Gigatiny using the Top-1 oracle-order hit rate. Each method uses its own retrieved demonstration set; the demonstrations are not necessarily shared across methods. The oracle ordering is therefore determined separately for each method and query, conditional on the demonstrations retrieved by that method.

For method $m$ and query $i$, let $\mathcal{C}_{i,m}$ denote the candidate orderings of its retrieved demonstrations, $s_{i,m}(\pi)$ the observed sentence-level ROUGE-1 score of ordering $\pi$, and $\hat{\pi}_{i,m}$ the ordering used by the method. The set of optimal orderings is defined as

$$\mathcal{O}_{i,m} = \arg\max_{\pi \in \mathcal{C}_{i,m}} s_{i,m}(\pi). \tag{16}$$

The Top-1 hit rate is computed as

$$\mathrm{Hit@1}(m) = \frac{100}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{\pi}_{i,m} \in \mathcal{O}_{i,m}], \tag{17}$$

where $N$ is the number of evaluated queries. Selecting any ordering tied for the highest task score counts as a hit.

Table 29 shows that both ADOICL variants achieve higher Top-1 hit rates than all compared baselines across the three LLMs. ADOICLE achieves 26.8%, 27.5%, and 23.6% on GLM, Llama, and Qwen, respectively, while ADOICLS achieves 28.0%, 27.8%, and 24.9%. The highest baseline hit rates are 6.5%, 6.2%, and 7.0% on the corresponding LLMs. ADOICLS therefore exceeds these baseline results by 21.5, 21.6, and 17.9 percentage points, respectively. These results show that both ADOICL variants more frequently select an optimal ordering within their respective retrieved demonstration sets in this evaluation.

This analysis characterizes ordering selection under each method's retrieval pipeline. Because the retrieved demonstrations and the number of tied optimal orderings may differ across methods, this comparison does not isolate ordering quality under a shared candidate set. It complements our controlled experiments with fixed demonstration sets and the downstream performance results.

### N.1 DIFFERENCES FROM PDDS

**Different optimization objectives.** PDDS optimizes which demonstrations are included in the final context, whereas ADOICL optimizes the presentation order of an already retrieved demonstration set. In ADOICL, all candidate contexts contain exactly the same demonstrations, preserving their contents and number. The decision concerns how these demonstrations should be arranged for a particular query and target LLM. This defines a distinct optimization problem: identifying an effective arrangement within a fixed set, rather than selecting the membership of that set.

**Different construction of training supervision.** PDDS constructs training contexts using the most similar single demonstration and the top-ranked group of demonstrations. ADOICL instead samples multiple distinct permutations of the same retrieved set for each training query and obtains a separate performance label for each permutation. The query, demonstration contents, and demonstration count remain fixed across these training examples. This construction provides supervision specifically for distinguishing alternative arrangements without changing which demonstrations are available. In the main setting, ADOICL samples five permutations per training query, providing ordering supervision without evaluating every possible arrangement with the target LLM.

**Explicit modeling of demonstration order.** PDDS describes the context as an unordered demonstration set in its problem formulation. ADOICL explicitly represents the arrangement of demonstrations through two implementations. ADOICLS encodes the complete demonstration sequence serialized in the candidate order. ADOICLE encodes individual demonstrations separately and concatenates their embeddings according to that order. In the latter implementation, each embedding's position in the concatenation identifies the corresponding demonstration's position in the context. These representations allow the predictor to distinguish candidate sequences containing identical demonstrations in different positions. This distinction concerns the published formulation and explicit treatment of order; it does not imply that PDDS's implementation is necessarily invariant to permutations.

**Different inference-time candidate construction.** PDDS first scores individual demonstrations, retains a ranked shortlist, and evaluates candidate contexts constructed through sliding windows over that shortlist. ADOICL first retrieves a fixed set and then constructs candidate permutations of its demonstrations. In the main four-demonstration setting, its predictor evaluates all twenty-four possible arrangements. Its candidate search therefore explores alternative placements of the same examples, rather than alternative subsets from a larger pool. Moreover, ranking individual demonstrations during preselection does not explicitly compare the performance of complete permutations. ADOICL performs this comparison through an ordering stage that preserves the retrieved demonstration membership.

**Representation reuse designed for permutation evaluation.** ADOICLE introduces a computation strategy tailored to evaluating multiple arrangements of identical demonstrations. It computes the query embedding and each demonstration embedding once, then constructs candidate representations by rearranging these cached embeddings. This design separates reusable text encoding from order-dependent prediction and avoids repeatedly encoding demonstration texts when evaluating different permutations. Each candidate still requires predictor evaluation, so the computational benefit concerns reduced encoding overhead rather than elimination of the permutation search. The design specifically exploits the unchanged demonstration contents across candidate arrangements.

**Different mechanism for many-shot contexts.** ADOICL introduces a group-level ordering extension for large demonstration sets. It clusters the retrieved demonstrations, represents each group through pooled embeddings, and evaluates alternative group arrangements. Within each group, demonstrations are arranged according to their similarity to the query. This mechanism preserves the retrieved examples while restricting the ordering search to arrangements of groups. It addresses the scalability of permutation optimization, whereas PDDS's described inference procedure constructs contexts through demonstration preselection and subset selection. The group-level extension searches a restricted family of arrangements and does not exhaustively optimize every possible individual-demonstration permutation.