Scaling In-Context Demonstrations with Structured Attention

Tianle Cai [*1] **Kaixuan Huang** [*1] **Jason D. Lee** [1] **Mengdi Wang** [1]

Abstract

The recent surge of large language models (LLMs) highlights their ability to perform in-context learning, i.e., "learning" to perform a task from a few demonstrations in the context without any parameter updates. However, their capabilities of in-context learning are limited by the model architecture: 1) the use of demonstrations is constrained by a maximum sentence length due to positional embeddings; 2) the quadratic complexity of attention hinders users from using more demonstrations efficiently; 3) LLMs are shown to be sensitive to the order of demonstrations. In this work, we tackle these challenges by proposing a better architectural design for in-context learning. We propose **SAICL** (**S**tructured **A**ttention for **I**n-**C**ontext **L**earning), which replaces the full-attention by a structured attention mechanism designed for in-context learning, and removes unnecessary dependencies between individual demonstrations, while making the model invariant to the permutation of demonstrations. We evaluate **SAICL** in a meta-training framework and show that **SAICL** achieves comparable or better performance than full attention while obtaining up to 3.4x inference speed-up. **SAICL** also consistently outperforms a strong Fusion-in-Decoder (FiD) baseline which processes each demonstration independently. Finally, thanks to its linear nature, we demonstrate that **SAICL** can easily scale to hundreds of demonstrations with continuous performance gains with scaling.

1. Introduction

Large language models (LLMs) have recently made notable advances with superior performance in downstream tasks (Brown et al., 2020; Chowdhery et al., 2022; Zhang et al., 2022; Hoffmann et al., 2022). One emergent property of LLMs is their ability to learn *in-context* (Brown et al., 2020), i.e., with a few task demonstrations provided in the prompt, LLMs are readily adapted to make accurate predictions without any parameter updates. The process of in-context learning only requires a single forward pass, making it an efficient and flexible alternative to fine-tuning for many real-world problems.

There are two major limitations of existing LLMs' in-context learning. *First*, it is expensive or infeasible to support a large number of demonstrations (typical choices ranging from 5 (Chung et al., 2022) to 32 (Brown et al., 2020)) due to the quadratic complexity of their attention mechanisms and the maximal sentence length constraints of the inputs. Therefore, whether scaling up the number of demonstrations will improve the performance of in-context learning remains unexplored. *Second*, the demonstrations are sequentially concatenated in the prompt, and recent evidence shows that performance is sensitive to the order of the demonstrations (Lu et al., 2022; Zhao et al., 2021). This artifact creates additional overhead and design for selecting a better order of demonstrations (Lu et al., 2022).

In this paper, we seek novel solutions to address the above challenges by *removing unnecessary dependencies between the demonstrations*, and *making the model invariant to their permutations*. This problem has yet to be comprehensively studied in the literature, while some existing ideas can be adapted. In particular, we find that the Fusion-in-Decoder (FiD) model (Izacard & Grave, 2021)[1]—originally proposed for open-domain question answering—is a strong baseline for in-context learning. In FiD, demonstration-input pairs are *independently* encoded and then concatenated before feeding into the decoder[2]. Independent encoding of demonstrations makes FiD scale *linearly* with the number of demonstrations, at the expense of fusing demonstrations only at the decoder stage. Another simple approach is *ensemble*, which averages the predictions based on individual demonstrations (Min et al., 2022a), and the dependencies between the demonstrations are completely ignored. Although both FiD and ensemble are already efficient and permutation-

[*]Equal contribution [1]Princeton University. Emails: {tianle.cai, kaixuanh, jasonlee, mengdiw}@princeton.edu

Work presented at the ES-FoMo Workshop at ICML 2023.

[1]A concurrent work (Ye et al., 2022a) also experiments with FiD for in-context learning. See more discussion in related work.

[2]FiD relies on the encoder-decoder Transformer architecture like T5 (Raffel et al., 2020).

invariant, we find that their performance is prone to saturate empirically as more demonstrations are used (Figure 4, 6(a)), and they often underperform standard concatenation-based baselines (Table 1, Appendix D). Therefore, how to strike the balance between efficient encoding and retaining necessary dependencies between demonstrations is the key research question to explore.

Motivated by the recent development of sparse Transformers (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020), we propose **SAICL** (**S**tructured **A**ttention for **I**n-**C**ontext **L**earning) as a replacement for the full-attention mechanism. Unlike existing sparse attentions that approximate the capability of full attention to process any input sequence, the design of **SAICL** is *tightly coupled with the structure of in-context learning*. As shown in Figure 1, we only allow the tokens of each demonstration to attend to themselves and the tokens of the test input, while the attentions of the tokens of the test input remain unchanged. In this way, the information from each demonstration is fused through the *global attention* of the test input in each attention layer and then sent back to all demonstrations in the next attention layer. Therefore, each demonstration is able to utilize the information from all other demonstrations while the complexity of the attention module is still *linear* to the number of demonstrations. Additionally, the positional encoding is only computed inside each demonstration, making the model invariant to the permutation of the demonstrations.

As **SAICL** requires modifications to the model architecture, we evaluate it in a meta-training framework (Min et al., 2021) for in-context learning. We use T5 (Raffel et al., 2020) as our base model and compare **SAICL** with standard T5 models with full attention and the FiD model (See Section 3.2 for a discussion about the choice of using T5). Specifically, we meta-train all models with the prompt being the same format of in-context learning using a set of source tasks and evaluate their performance when transferred to unseen target tasks. Our experiments (Section 5.1) indicate that **SAICL** often achieves better performance than FiD (18 wins out of 28 settings) and is able to match or even beat (13 wins out of 28 settings) the full attention baseline. Furthermore, we demonstrate that **SAICL** can scale to hundreds of demonstrations, and improve its performance consistently with more demonstrations (Figure 4). Thanks to its linear complexity, **SAICL** also achieves significant speed-ups during inference—up to 3.4x for 64 demonstrations and 6.3x for 128 demonstrations.

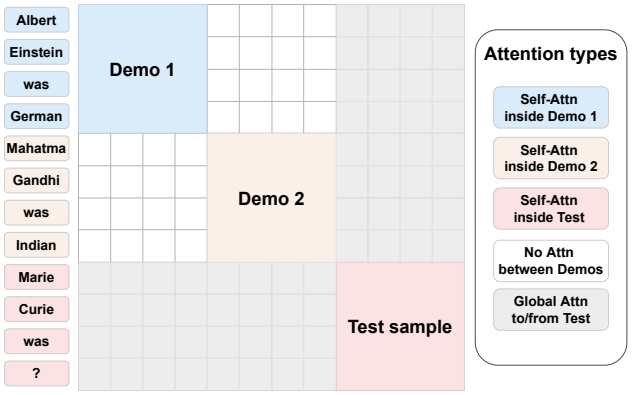


Figure 1. An illustration of **SAICL** with two demonstrations. **SAICL** is a structured attention mechanism designed for in-context learning that can directly replace the self-attention layers in Transformer models. By removing the attention across different demonstrations while keeping the test input to attend to all tokens globally, we reduce the redundancy in computation but retain necessary dependencies between different demonstrations. **SAICL**, therefore, 1) enjoys linear computational and memory complexity; 2) has no limitation on the number of demonstrations used; 3) is invariant to the permutation of the demonstrations. `Demo` i stands for the i-th demonstration, `test` stands for the test input.

2. Related Work

In-context learning. Large language models can perform in-context learning (Brown et al., 2020), where LLMs are adapted to new tasks by conditioning them on a few demonstrations. Although in-context learning is shown mainly with decoder-only models such as GPTs(Radford et al., 2019; Brown et al., 2020), several recent works also experiment with encoder-decoder models such as T5 (Patel et al., 2022; Chung et al., 2022; Tay et al., 2022b).

Many recent works aim to understand how in-context learning works through empirical and theoretical investigation (Xie et al., 2021; Min et al., 2022b; Akyürek et al., 2022; Garg et al., 2022; von Oswald et al., 2022; Dai et al., 2022; Chan et al., 2022; Lyu et al., 2022; Qiao et al., 2022; Li et al., 2023). Some other studies focus on improving the performance of in-context learning through calibration (Zhao et al., 2021; Holtzman et al., 2021; Min et al., 2022a), self-supervised training (Chen et al., 2022a), and meta-learning (Min et al., 2021; Chung et al., 2022). Other directions include retrieving demonstrations that are semantically similar to the test input (Liu et al., 2022b; Rubin et al., 2021), or using chain-of-thought prompts to enhance reasoning (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022b;a).

One notorious property of in-context learning is its sensitivity to the order of demonstrations. Zhao et al. (2021) show that LLMs are prone to outputting the same label as the last demonstration (aka recency bias). Lu et al. (2022) find that under some permutation of the demonstrations, the performance is close to random guesses, and this phenomenon exists across various models. As a remedy, they propose to exhaustively search for all possible prompt orders and use

entropy statistics to select the best order. The permutation sensitivity of in-context learning incurs a computational overhead to search for a good prompt order. This motivates us to address the order-sensitivity issue of in-context learning through a better model design.

Efficient Transformers. Despite the success of the Transformer architecture (Vaswani et al., 2017) in a wide range of machine learning applications, its computation and memory complexity scale quadratically with the input length. This makes it intractable for modeling long-range contexts. Therefore, a variety of efficient Transformer models have been proposed to mitigate this issue (Child et al., 2019; Beltagy et al., 2020; Wang et al., 2020; Kitaev et al., 2019; Katharopoulos et al., 2020; Zaheer et al., 2020; Qiu et al., 2020; Tay et al., 2020; Peng et al., 2021; Qin et al., 2021; Zhou et al., 2021). We refer the readers to Tay et al. (2022a) for a comprehensive survey. In this paper, we draw inspiration from the design of sparse attention mechanisms in efficient Transformers (Beltagy et al., 2020; Zaheer et al., 2020). However, instead of using sparse attention to approximate the ability of full attention to process any sequences, we exploit the inherent structure of in-context learning (test input follows a number of demonstrations), and design a structured attention mechanism tailored for the problem.

Comparison with concurrent works. Concurrent with our work, there are several works (Hao et al., 2022; Ratner et al., 2022; Ye et al., 2022a) exploring architectural designs to improve the in-context learning ability of LLMs. Among them, Hao et al. (2022); Ratner et al. (2022) focus on tackling the limitation of context length by independently processing several prompts *within the max length constraint* and then combining them. This idea is similar to our adaptation of FiD while Hao et al. (2022); Ratner et al. (2022) use *fixed schemes* (e.g., average) for fusing parallel prompts so that their methods can be used directly on top of pre-trained models without further fine-tuning. Compared to their methods, we, in addition, seek to address the efficiency and instability issues while exploring better encoding to retain the dependencies between demonstrations. These goals cannot be directly achieved by processing several grouped prompts in parallel. Ye et al. (2022a) investigate different ways of fusing demonstrations, and they also observe that adapting FiD to in-context learning is effective (usually better than the ensemble-based method and more efficient than the concatenation-based method), which is aligned with our findings (Figure 6(a) and Section 5.1).

Due to length limitation, we refer the readers to Appendix E for more related works on (few-shot) fine-tuning.

3. Efficient In-Context Learning with SAICL

3.1. Desiderata of In-Context Learning Models

For in-context learning, assume we are given k demonstrations $(\mathbf{x}_i, y_i)_{i=1}^k$, we construct the prompt as the concatenation of all the demonstration pairs and the test input: $\mathbf{x}_{\text{prompt}} = (\mathbf{x}_1, y_1, \mathbf{x}_2, y_2, \ldots, \mathbf{x}_k, y_k, \mathbf{x}_{\text{test}})$. When we evaluate the model on a fresh input $\mathbf{x}_{\text{test}}$ to predict from a set of candidate answers $Y_{\text{test}} = \{y_{\text{test}}^{(1)}, \ldots, y_{\text{test}}^{(m)}\}$, we condition the language model on $\mathbf{x}_{\text{prompt}}$ and query the probabilities of its continuations to inference about the answer $p(y_{\text{test}} \mid \mathbf{x}_{\text{prompt}})$ for each $y_{\text{test}} \in Y_{\text{test}}$. [3] Afterwards, the answer for $\mathbf{x}_{\text{test}}$ is $\text{argmax}_{y_{\text{test}}} p(y_{\text{test}} \mid \mathbf{x}_{\text{prompt}})$. For encoder-decoder models such as T5, the $\mathbf{x}_{\text{prompt}}$ is fed into the encoder, whose embedding is then computed as key-value pairs for the cross-attention modules of the decoder to perform auto-regressive decoding.

We state the high-level design goals for an ideal in-context learner. 1) **Extensibility.** The model should be capable of using many demonstrations to boost performance without any *hard-coded constraints* on the number of demonstrations caused by the model design, e.g., the max length constraints due to the limited number of positional encodings. 2) **Efficiency.** The model should be able to *efficiently* scale to a large number (e.g., hundreds) of demonstrations. Ideally, the computational complexity should only have a linear dependency on the number of demonstrations. 3) **Invariance.** The model should be invariant to the permutation of demonstrations, which saves extra computation on searching over the *combinatorial space of possible permutations* and preserving the natural symmetry of demonstrations.

3.2. SAICL

We propose **S**tructured **A**ttention for **I**n-**C**ontext **L**earning (**SAICL**, pronounced as *cycle*) that replaces the full bidirectional attention in the *encoder* of encoder-decoder models such as T5. In **SAICL**, we exploit the special structure of the prompts of in-context learning. We observe that the computational burden comes from the dense attention between *all pairs of demonstrations*. Intuitively, the understanding of each demonstration $(\mathbf{x}_i, y_i)$ only *loosely* depends on the understanding of all other demonstrations. Therefore, in **SAICL**, we remove the attentions across different demonstrations as shown in Figure 1. Meanwhile, we keep the attention from $\mathbf{x}_{\text{test}}$ global to make $\mathbf{x}_{\text{test}}$ an aggregator of information from different demonstrations. Concretely, the tokens of the i-th demonstration $(\mathbf{x}_i, y_i)$ can only attend to 1) tokens within the same demonstration and 2) the test tokens, while the test tokens $\mathbf{x}_{\text{test}}$ can attend to all the to-

[3] We use the *direct* method for demonstration in this section. See Section 4.1 for the comparison between the *direct* method and the *channel* method.

kens. This way, the test input tokens can fuse information from all demonstrations and send information back to each demonstration in the next layer, which implicitly makes each demonstration fully utilize information from all other demonstration exemplars.

For simplicity of illustration, we assume each sample only has *one token* and ignore the relative positional encodings (for a comprehensive description of **SAICL**, please refer to the pseudo-code in Appendix B). Let $\{(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^k$ and $(\mathbf{q}_{\text{test}}, \mathbf{k}_{\text{test}}, \mathbf{v}_{\text{test}})$ be the (query, key, value) triplets of demonstrations and test sample for calculating the attention, respectively. Then the output of **SAICL** at position $i \in [1, k]$ is calculated as:

$$\mathbf{z}_i = \frac{\exp\left(\mathbf{q}_i^\top \mathbf{k}_i\right) \mathbf{v}_i + \exp\left(\mathbf{q}_i^\top \mathbf{k}_{\text{test}}\right) \mathbf{v}_{\text{test}}}{\exp\left(\mathbf{q}_i^\top \mathbf{k}_i\right) + \exp\left(\mathbf{q}_i^\top \mathbf{k}_{\text{test}}\right)};$$

and the output at the test sample position is

$$\mathbf{z}_{\text{test}} = \frac{\sum_{i=1}^k \exp\left(\mathbf{q}_{\text{test}}^\top \mathbf{k}_i\right) \mathbf{v}_i + \exp\left(\mathbf{q}_{\text{test}}^\top \mathbf{k}_{\text{test}}\right) \mathbf{v}_{\text{test}}}{\sum_{i=1}^k \exp\left(\mathbf{q}_{\text{test}}^\top \mathbf{k}_i\right) + \exp\left(\mathbf{q}_{\text{test}}^\top \mathbf{k}_{\text{test}}\right)}$$

Efficiency. This sparse structure of **SAICL** then makes the computational complexity and the memory consumption only *linearly* depend on the number of demonstrations. Concretely, assume there are k demonstrations, and each demonstration and test sample has a max length L, then the computational and memory complexity of full attention will be $\mathcal{O}(k^2L^2)$ while the complexity of **SAICL** will be $\mathcal{O}(kL^2)$. As shown in Figure 2, we create fake inputs where the number of tokens in each demonstration is fixed to 64 and 128 for evaluating the scalability of inference time of **SAICL** and full attention. The reported inference times are averaged across ten runs tested on RTX A6000 GPUs. We observe that **SAICL** scales much better than the full attention, yet the advantage of **SAICL** is less significant when the number of demonstrations is small. We hypothesize that this is mainly due to the implementation overhead of **SAICL**, which may be further improved.

Extensibility and permutation invariance. Unlike the decoder-only GPT models, the T5 encoder only relies on the relative positional encodings to determine the order of the tokens. Specifically, instead of adding an absolute positional encoding to each token embedding, T5 adds a bias term to the pre-softmax value of the attention heads, which only depends on the difference between the query position and the key position. In **SAICL**, we only keep the relative positional encodings inside the tokens of each demonstration or test sample. Therefore, the max length constraint only limits the length of each sample, which is usually short in practice. Also, in T5, the order of demonstrations is distinguished solely via the relative positional encodings

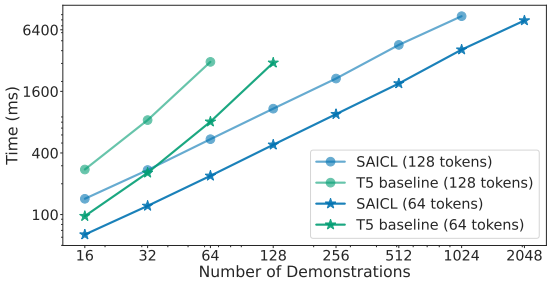


Figure 2. The inference times (ms) v.s. different numbers of demonstrations. The inference time of **SAICL** consistently scales better than the full attention baseline with different per sample length $L \in \{64, 128\}$. By design, the computational complexity of **SAICL** scales linearly to the number of demonstrations k while the full attention scales quadratically ($\mathcal{O}(kL^2)$ v.s. $\mathcal{O}(k^2L^2)$). Also note that the T5 baseline runs out of memory quickly (64 shots), even on an A6000 with 50 GB of memory.

in the attentions *across* difference demonstrations. Thus, after removing these attentions, our **SAICL** automatically achieves the permutation invariance goal.

Discussion on the choice of encoder-decoder model. We build **SAICL** and conduct experiments on top of the T5 model, and many ideas benefit from the T5 architecture: 1) **More flexible information exchange.** The bidirectional attention mechanism enables free information exchange between different demonstrations. Specifically, in **SAICL**, the information can be passed through the global attention by the test sample. Also, the encoder-decoder separation enables parallel encoding in FiD. 2) **Easier to keep the symmetry.** It is more natural to use bidirectional attention for encoding demonstrations because there is no canonical order of demonstrations, but the causal mask in decoder-only models induces an order. Also, as mentioned before, the relative positional encoding in T5 enables the permutation invariance in **SAICL**. Yet, we believe that it is easy to adapt our methods to a decoder-only model with modifications on the positional encoding and the attention mask.

3.3. Meta-Training

We use meta-training to fine-tune the models following Meta In-Context Learning (MetaICL) (Min et al., 2021). The meta-training objective is

$$\max_\theta \hat{\mathbb{E}} \log p_\theta(y_{\text{test}} \mid \mathbf{x}_1, y_1, \ldots, \mathbf{x}_k, y_k, \mathbf{x}_{\text{test}}),$$

where the model is trained using a mixture of source tasks and the prompt has the same format of in-context learning, i.e., the concatenation of the demonstrations and the test input. During training, we first sample a source task and

then randomly draw $(\mathbf{x}_i, y_i)$'s and $(\mathbf{x}_{\text{test}}, y_{\text{test}})$ from its training dataset.

We use the same datasets, evaluation protocols, and prompt template designs as MetaICL. Concretely, we use 142 tasks from CrossFit (Ye et al., 2021) and UnifiedQA (Khashabi et al., 2020), which include text classification, question answering (QA), natural language inference (NLI), and paraphrase detection. They are divided into seven transfer settings, each consisting of a non-overlapping pair of source and target tasks. A subset of the target tasks is from completely different domains from the training tasks, e.g., medical, financial, and climate. Following Min et al. (2021), we highlight these tasks as unseen domain tasks, where the gains of in-context learning over the multi-task fine-tuning setting are more significant.

At test time, we evaluate the meta-trained model on the unseen target tasks with k-shot demonstrations and average the results across 5 different random sets of the demonstrations. Macro-F1 and accuracy are used as evaluation metrics for classification tasks and non-classification tasks, respectively. In order to investigate the scaling behavior of in-context learning, for each transfer case and each method, we meta-train two models with different training-time numbers of demonstrations: train $k = 16$ and train $k = 64$. Then, we use the corresponding fine-tuned model at test time and evaluate it under test $k = 16$ and test $k = 64$ settings. In ablation studies, we use the high-resource to low-resource transfer setting (HR→LR) following Min et al. (2021). This setting is the most representative one because its source and target both cover all the task types.

4. Experimental Setup

4.1. Settings

Base models. Different from MetaICL (Min et al., 2021), we choose T5-LM-adapt large (770M parameters)[4] as our base model, which is additionally fine-tuned from T5.1.1 using LM objective[5]. Please see Section 3.2 for a discussion about the choice of using T5.

Direct v.s. channel. There are two mainstream methods to perform in-context learning: direct method and channel method (Min et al., 2022a). For *direct* method, the LM is prompted with $(\mathbf{x}_1, y_1, \ldots, \mathbf{x}_k, y_k, \mathbf{x}_{\text{test}})$ to predict the probability of the answer y_{test}. For *channel* method, the LM is prompted with $(y_1, \mathbf{x}_1, \ldots, y_k, \mathbf{x}_k, y_{\text{test}})$ to predict the probability of the test input $\mathbf{x}_{\text{test}}$. As suggested by Min et al. (2022a; 2021), the channel method usually performs better than the direct method thanks to better calibration, so we choose to use the *channel* method in *all* our experiments.

For detailed data processing and optimization settings, please see Appendix C.

4.2. Baselines

We compare **SAICL** with the following baseline methods.

T5 baseline. We meta-train the LM-adapted T5-large model without modifying its attention mechanism using the channel method. In this case, the computational complexity is $\mathcal{O}(k^2)$.

Fusion-in-Decoder (FiD) (Izacard & Grave, 2021). We adapt the FiD to in-context learning by independently passing each $\mathbf{x}_{\text{prompt},i} = (y_i, \mathbf{x}_i, y_{\text{test}})$ through the standard T5 encoder and concatenate all their outputs, and then feed the concatenation into the cross-attention layers of the T5 decoder to calculate the conditional probability of x_{test}. The computational complexity is $\mathcal{O}(k)$ for FiD.

MetaICL baselines. We report four baseline results from the original MetaICL paper (Min et al., 2021) where *GPT-2 large models* are used:

- Multi-task 0-shot: Meta-trained without any demonstration using *direct* method.
- Channel Multi-task 0-shot: Meta-trained without any demonstration using *channel* method.
- MetaICL: Meta-trained with $k = 16$ demonstrations using *direct* method.
- Channel MetaICL: Meta-trained with $k = 16$ demonstrations using *channel* method.

5. Experiments

5.1. Main Results

We present the results in Table 1 and summarize the findings below.

Our baselines (T5 and FiD) are strong. Although T5 large is roughly the same size as GPT-2 large, our experimental results indicate that T5 baselines generally perform better than GPT-2 in the Meta-ICL setting (9 wins out of 14 settings). We hypothesize that the benefits partially come from the bidirectional attention of the T5 encoder, which, compared to decoder-only models such as GPT, allows each demonstration to utilize the information from all other demonstrations. For FiD, our experiments show that it can be seamlessly adapted to in-context learning and usually match the performance of full attention without any fusion of demonstrations in the encoder.

[4] https://huggingface.co/google/t5-large-lm-adapt

[5] This LM-adapted version of T5 is suggested to have better in-context learning ability than the original ones (Sanh et al., 2021).

Method	Complexity	HR→LR	Class →Class	non-Class →Class	QA →QA	non-QA →QA	non-NLI →NLI	non-Para →Para
All target tasks								
Multi-task 0-shot[1]	$\mathcal{O}(1)$	35.6	37.3	36.8	45.7	36.0	40.7	30.6
Channel Multi-task 0-shot[1]		38.8	40.9	42.2	42.1	36.4	36.8	35.1
MetaICL[1]	$\mathcal{O}(k^2)$	43.3	43.4	38.1	46.0	38.5	49.0	33.1
Channel MetaICL[1]		49.1	50.7	50.6	44.9	41.9	**54.6**	52.2
T5 baseline		50.8	51.9	54.7	46.2	44.5	45.7	51.3
T5 baseline (64-shot)		52.7	57.2	**58.9**	46.4	44.6	48.1	55.6
FiD	$\mathcal{O}(k)$	49.0	50.1	50.7	45.7	44.2	48.1	51.7
SAICL		49.8	51.0	50.9	45.7	43.8	44.8	51.8
FiD (64-shot)		50.7	57.6	48.6	**46.6**	**45.3**	53.7	57.2
SAICL (64-shot)		**53.6**	**58.3**	53.3	46.3	44.3	48.9	**57.8**
Target tasks in unseen domains								
Multi-task 0-shot[1]	$\mathcal{O}(1)$	35.4	28.0	28.6	**71.2**	40.3	33.5	35.0
Channel Multi-task 0-shot[1]		36.3	31.1	34.3	54.4	39.4	50.8	34.1
MetaICL[1]	$\mathcal{O}(k^2)$	35.3	32.3	28.1	69.9	48.3	**80.1**	34.0
Channel MetaICL[1]		47.7	41.9	48.0	57.9	47.2	62.0	51.0
T5 baseline		53.2	50.1	50.5	50.8	50.8	60.0	47.4
T5 baseline (64-shot)		54.7	54.6	**54.0**	53.3	46.1	62.3	53.8
FiD	$\mathcal{O}(k)$	49.8	47.3	47.3	51.8	**53.3**	59.8	50.9
SAICL		49.0	52.8	47.9	52.8	50.4	57.5	53.8
FiD (64-shot)		55.3	55.1	40.7	53.3	48.3	67.6	58.3
SAICL (64-shot)		**57.4**	**56.8**	48.9	52.9	49.4	63.4	**60.8**

Table 1. Main results. *We use 16 shots by default.* For the same number of demonstrations, we see that **SAICL** performs better than FiD in 18 out of 28 settings, and is able to achieve comparable performance as the T5 baseline. Scaling up the number of demonstrations k further boosts the performance for T5 baseline, FiD, and **SAICL**. [1]The results are copied from Min et al. (2021), where the base model used for these 4 baselines is *GPT-2 large*. Our result indicates that the T5 baseline performs better than GPT-2. **Bold** indicates the best result in each column. The results are averaged across 5 different samples of demonstrations.

SAICL enables sufficient fusion among demonstrations. With the same number of demonstrations, we observe that **SAICL** usually performs better than FiD (18 wins out of 28 settings) while maintaining the same linear complexity. Analogously, we can view our method as "fusion in encoder", which has more flexibility for exchanging information between demonstrations. Compared to the full attention, **SAICL** can usually match or even beat its performance while being much more efficient. These results indicate that **SAICL** enables sufficient fusion among the demonstrations as the full attention, while completely disabling information exchange in the encoder (FiD) will lead to worse performance.

Scaling up the number of demonstrations k boosts the performance. Our results show that increasing k from 16 to 64 can significantly improve the performance for all models (by $\sim 3\%$ in average). This observation is seemingly in contrast with the finding in the MetaICL paper (Min et al., 2021), where the authors show the performance improvement tends to saturate after 32 demonstrations. We hypothesize that the difference is because the 1024 length limitation of the GPT-2 model used in Min et al. (2021) constrains the effective context. This finding suggests that the potential of boosting performance with more demonstrations might be underestimated because of inappropriate architectural design.

Different behaviors on various types of tasks. We notice that the methods we test behave differently across different tasks, and no one can rule over all tasks. This is because solving different tasks requires different forms of information. For example, to solve classification tasks with in-context learning, it is crucial to determine the label space and the format by which a classification problem is converted to natural language. In Figure 3, we inspect the behaviors of **SAICL** and FiD on different types of tasks under the *test* domain in the HR→LR setting, where all types of tasks are covered by both training and test domains. One observation is that the benefits of **SAICL** are more pronounced on classification tasks. We hypothesize that this is because classification tasks require more subtle knowledge about the format and label space, which benefits from the fusion of demonstrations enabled by **SAICL** in the encoder. This observation also coincides with the finding in Min et al. (2022b) that in-context learning relies on the label space, input format, and distribution of input text to work.

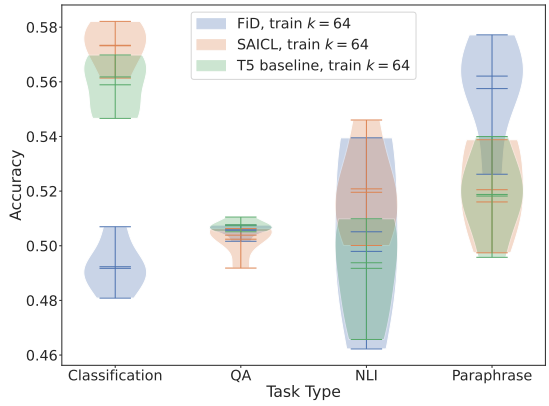


Figure 3. Comparison of **SAICL** and FiD across task types. We group the performance of **SAICL** and FiD in terms of task type in the HR→ LR setting. The plot shows that the benefits of **SAICL** are more pronounced on classification tasks. We hypothesize that classification tasks rely more on the input format and label space, which can be extracted by comparing different demonstrations, and thus, more dependencies are needed.

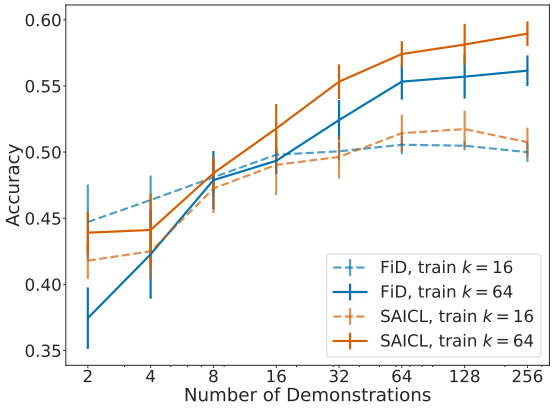


Figure 4. Scaling behaviour of FiD and **SAICL**. Generally, the performances of both **SAICL** and FiD improve when the number of demonstrations scales up. **SAICL** achieves significantly better performance when using many demonstrations. A larger train k is beneficial when extrapolating to larger test k. The reported performances are tested on HR→ LR *unseen domain* setting.

Discussion on the model size. Due to computational limitations, we use the T5 large model (770M) in this paper. It is an interesting future work to extend our methods to larger models. We notice that in the concurrent work Ye et al. (2022a) (Figure 2), the authors already find that the dominance of FiD compared to ensemble methods is much more significant when the model size increases from T5 large to T5 XL (3B). We believe that as we scale up the model size, the advantage of **SAICL** and FiD will keep increasing.

5.2. Scaling to Hundreds of Demonstrations with SAICL

In Section 5.1, we show that raising the number of demonstrations k from 16 to 64 significantly improves the performance, showcasing the potential of increasing demonstrations. In this section, we further investigate the performance of **SAICL** and FiD when scaling up to *hundreds* of demonstrations (T5 baseline will run out of GPU memory in these settings). As usual, we use HR→ LR setting as our testbed. Due to computational limitations, we focus on the *unseen domain* setting where fewer tasks are evaluated. Here are several observations from the scaling curves in Figure 4.

Performance can be further improved when we scale up to hundreds of demonstrations. By scaling up the number of demonstrations, we can further boost the performance of both **SAICL** and FiD. The results also show that when the number of demonstrations is large, the benefit of **SAICL** is more prominent, suggesting the importance of enabling the fusion among demonstrations in the encoder.

Training with larger k can enhance the scalability. We also explore the influence of the training-time number of demonstrations (train k). We observe that for both FiD and **SAICL**, train $k = 64$ achieves better performance than train $k = 16$ when the test $k \geq 16$. This suggests that a larger training-time number of demonstrations is beneficial for extrapolating to a larger test-time number of demonstrations. It is an interesting future direction to explore how to improve the model's ability to *extrapolate* to larger k.

5.3. Combining with Ensemble

Ensemble methods are commonly used in machine learning. For in-context learning, Min et al. (2022a); Ye et al. (2022a) show that simply aggregating k predictions based on k single-shot prompts is already an effective approach to achieve extensibility, efficiency, and permutation invariance. However, as shown in Ye et al. (2022a) and our experiments in Appendix D, the ensemble of single-shot predictions usually under-performs FiD and also saturates fast as k grows. Therefore, in this section, we investigate the hybrid approach of combining ensemble and *few-shot*-prompted models with the hope of achieving better performance at the expense of *losing efficiency and permutation invariance*. Concretely, given k demonstrations, we equally split them into G groups, independently construct the prompts, make predictions over each group, and finally average the logits of all groups to obtain the final prediction.

We run experiments over combinations of $k \in$

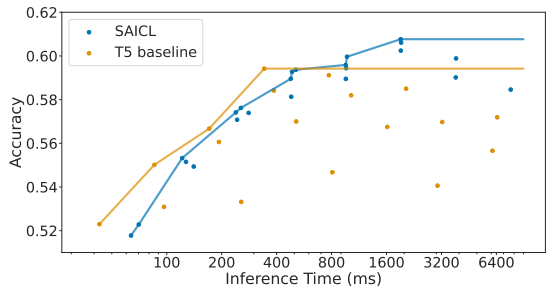


Figure 5. Pareto frontier for accuracy v.s. inference time. We vary the ensemble size in $\{1, 2, 4, 8\}$ and the number of demonstrations in $\{16, 32, 64, 128, 256, 512\}$ for T5 baseline and **SAICL**. The baseline dominates when the inference time is short. But **SAICL** performs better when using more demonstrations. The accuracies are tested on the HR→LR *unseen domain* setting.

$\{16, 32, \ldots, 512\}$ and $G \in \{1, 2, 4, 8\}$, and compare the baseline T5 model and **SAICL** under HR→LR unseen domain setting. We plot the Pareto frontier of the performance versus the inference time in Figure 5. As we can see, combining with the ensemble method can further boost the performance of both baseline and **SAICL**. Remarkably, when $k = 512$ and $G = 8$, **SAICL** achieves 60.8% accuracy, which is 11.8% higher than the 16-shot **SAICL** baseline, demonstrating the great potential of using more demonstrations. Meanwhile, the performance of the baseline model is also boosted, where the best 59.4% accuracy is achieved when $k = 64$ and $G = 8$. Generally, for the baseline model, the improvement is more significant when each group is reasonably small (e.g., 8 demonstrations). And since when each prompt only has a few demonstrations, the efficiency advantage of **SAICL** is less significant (as is shown in Figure 2), we see in the Pareto frontier curve that the baseline dominates when inference is fast. Yet, **SAICL** has a better frontier when the inference time is greater than 800 ms, i.e., using more demonstrations.

6. Discussion

This paper takes an initial step toward a better model design for in-context learning, yet many interesting questions remain open. We discuss the important ones below and summarize other discussion questions in Appendix A.

The applicability of SAICL. In this paper, we use meta-training to fine-tune the pre-trained model due to the modification of the attention mechanism. Other potential ways to apply the idea of **SAICL** include 1) approximating the full attention with **SAICL**, and then directly using pre-trained models without further fine-tuning. This requires careful alignment of the positional encoding; 2) designing unsupervised objectives to pre-train/fine-tune a **SAICL** model. A possible approach is to use an additional retrieval module to find similar sentences as demonstrations and use the language modeling objective on the test sample.

Extrapolation ability to more demonstrations. In our experiments of scaling up the number of demonstrations in Section 5.2, we observe that, although **SAICL** removes the restriction on the number of demonstrations, the effect of increasing the number of demonstrations k saturates quickly when k is larger than the one used in training time. It is an interesting future direction to explore how to improve the ability of the model to extrapolate to larger k.

Combination of different methods. Following the discussion of extrapolation ability in Section 5.2, one possible way to extrapolate to larger k is to combine different fusion methods. The combination with the ensemble methods in Section 5.3 is an example of the idea. Moreover, the methods of Hao et al. (2022); Ratner et al. (2022) may also be used to further boost the performance when there are many demonstrations available (see our tentative experimental results in Appendix D). An interesting open question will be *how to combine different methods to achieve the best performance, given the number of available demonstrations*.

Understanding in-context learning. From our experimental results, we see that removing the attention between different demonstrations does not hurt the performance of in-context learning in most settings. Moreover, fusion only in the decoder already suffices to achieve comparable performance to the full attention model. This observation raises the question of *how much fusion of information is needed for in-context learning?*. Furthermore, we find different tasks may have different degrees of requirements on the fusion of information. For example, classification tasks benefit more from the information exchange between demonstrations, as discussed in Section 5.1.

7. Conclusion

In this paper, we seek to improve the architecture design of large language models for in-context learning. We propose **SAICL**, a structured attention mechanism that exploits the prompt structure of in-context learning. The computational and memory complexity of **SAICL** only scales linearly with the number of demonstrations and the model does not have hard-coded restrictions on the number of demonstrations. Furthermore, our method is completely insensitive to the permutations of the demonstrations.

We adapt the FiD method from open-book question answering to in-context learning, which also enjoys extensibility, efficiency, and permutation invariance as our method. We

empirically compare our method against the T5 baseline and FiD under the same setting as (Min et al., 2021). Our experiments demonstrate that FiD is a strong baseline, yet **SAICL** outperforms FiD under 18 of 28 settings and is able to match or beat the performance of the T5 baseline (13 wins out of 28 settings). Then, we show that **SAICL** is able to efficiently scale up to hundreds of demonstrations (6.3x speed-up when using 128 demonstrations compared to the baseline) and can continuously boost test performance with more demonstrations (e.g., 13.8% relative performance improvement when we increase k from 16 to 256 under HR→LR unseen domain setting).

Acknowledgements

Tianle and Kaixuan would like to thank Professor Danqi Chen for wonderful discussions and valuable suggestions. JDL acknowledges support of the ARO under MURI Award W911NF-11-1-0304, the Sloan Research Fellowship, NSF CCF 2002272, NSF IIS 2107304, NSF CIF 2212262, ONR Young Investigator Award, and NSF CAREER Award 2144994. Mengdi Wang acknowledges the support by NSF grants DMS-1953686, IIS-2107304, CMMI-1653435, ONR grant 1006977, and C3.AI.

References

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

An, S., Li, Y., Lin, Z., Liu, Q., Chen, B., Fu, Q., Chen, W., Zheng, N., and Lou, J.-G. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*, 2022.

Bapna, A. and Firat, O. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1538–1548, 2019.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chan, S. C., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., McClelland, J., and Hill, F. Data distributional properties drive emergent few-shot learning in transformers. *arXiv preprint arXiv:2205.05055*, 2022.

Chen, M., Du, J., Pasunuru, R., Mihaylov, T., Iyer, S., Stoyanov, V., and Kozareva, Z. Improving in-context few-shot learning via self-supervised training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3558–3573, Seattle, United States, July 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.260. URL https://aclanthology.org/2022.naacl-main.260.

Chen, Y., Liu, Y., Dong, L., Wang, S., Zhu, C., Zeng, M., and Zhang, Y. Adaprompt: Adaptive model training for prompt-based nlp. *arXiv preprint arXiv:2202.04824*, 2022b.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., and Wei, F. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3816–3830, 2021.

Garg, S., Tsipras, D., Liang, P., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Guo, D., Rush, A. M., and Kim, Y. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4884–4896, 2021.

Hao, Y., Sun, Y., Dong, L., Han, Z., Gu, Y., and Wei, F. Structured prompting: Scaling in-context learning to 1,000 examples. *arXiv preprint arXiv:2212.06713*, 2022.

Henderson, J., Ruder, S., et al. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, 2021.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Holtzman, A., West, P., Shwartz, V., Choi, Y., and Zettlemoyer, L. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7038–7051, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.564. URL https://aclanthology.org/2021.emnlp-main.564.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021.

Izacard, G. and Grave, É. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, 2021.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., and Hajishirzi, H. Unifiedqa: Crossing format boundaries with a single qa system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1896–1907, 2020.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and implicit model selection in in-context learning. *arXiv preprint arXiv:2301.07067*, 2023.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*, 2022a.

Liu, J., Shen, D., Zhang, Y., Dolan, W. B., Carin, L., and Chen, W. 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, 2022b.

Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., and Tang, J. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. 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, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556.

Lyu, X., Min, S., Beltagy, I., Zettlemoyer, L., and Hajishirzi, H. Z-icl: Zero-shot in-context learning with pseudo-demonstrations. *arXiv preprint arXiv:2212.09865*, 2022.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

Min, S., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5316–5330, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.365. URL https://aclanthology.org/2022.acl-long.365.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022b.

Patel, A., Li, B., Rasooli, M. S., Constant, N., Raffel, C., and Callison-Burch, C. Bidirectional language models are also few-shot learners. *arXiv preprint arXiv:2209.14500*, 2022.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N., and Kong, L. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., and Chen, H. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*, 2022.

Qin, Z., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., Yan, J., Kong, L., and Zhong, Y. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2021.

Qiu, J., Ma, H., Levy, O., Yih, W.-t., Wang, S., and Tang, J. Blockwise self-attention for long document understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pp. 2555–2565, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Ratner, N., Levine, Y., Belinkov, Y., Ram, O., Abend, O., Karpas, E., Shashua, A., Leyton-Brown, K., and Shoham, Y. Parallel context windows improve in-context learning of large language models. *arXiv preprint arXiv:2212.10947*, 2022.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Rubin, O., Herzig, J., and Berant, J. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2021.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Sung, Y.-L., Nair, V., and Raffel, C. A. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Tay, Y., Bahri, D., Metzler, D., Juan, D., Zhao, Z., and Zheng, C. Synthesizer: Rethinking self-attention in transformer models. arxiv 2020. *arXiv preprint arXiv:2005.00743*, 2, 2020.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6): 1–28, 2022a.

Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Bahri, D., Schuster, T., Zheng, H. S., Houlsby, N., and Metzler, D. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*, 2022.

Wang, B., Min, S., Deng, X., Shen, J., Wu, Y., Zettlemoyer, L., and Sun, H. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022a.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity, 2020.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2021.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2021.

Ye, Q., Lin, B. Y., and Ren, X. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7163–7189, 2021.

Ye, Q., Beltagy, I., Peters, M. E., Ren, X., and Hajishirzi, H. Investigating fusion methods for in-context learning. preprint under review, 2022a.

Ye, S., Kim, D., Jang, J., Shin, J., and Seo, M. Guess the instruction! flipped learning makes language models stronger zero-shot learners. *arXiv preprint arXiv: Arxiv-2210.02969*, 2022b.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

Zaken, E. B., Goldberg, Y., and Ravfogel, S. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pp. 12697–12706. PMLR, 2021.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021.

A. More Discussion

We summarize several discussion questions about this paper and gather the related discussions in the main paper here for better understanding.

Why use an encoder-decoder model (T5)? We discuss this in Section 3. The core ideas are pasted here: 1) More flexible information exchange. The bidirectional attention mechanism enables free information exchange between different demonstrations. Specifically, in **SAICL**, the information can be passed through the global attention by the test sample. Also, the encoder-decoder separation enables parallel encoding in FiD. 2) Easier to keep the symmetry. It is more natural to use bidirectional attention for encoding demonstrations because there is no canonical order of demonstrations, but the causal mask in decoder-only models induces an order. Also, as mentioned before, the relative positional encoding in T5 enables the permutation invariance in **SAICL**. Yet, we believe that it is easy to adapt our methods to a decoder-only model with modifications on the positional encoding and the attention mask.

Why does permutation invariance matter? We discuss this in Section E and 3. In summary, this is because 1) The performance is sensitive to the ordering (Zhao et al., 2021; Lu et al., 2022). This makes the results unstable and fixing this requires additional computational costs for searching for a good ordering. 2) The demonstrations are naturally symmetric and the model should fit into this symmetry.

What will happen when scaling up the model size? We discuss this in Section 5.1. We believe it is an interesting future work to extend our methods to larger models. Also, we notice that in the concurrent work Ye et al. (2022a) (Figure 2), the authors already find that the dominance of FiD compared to ensemble methods is much more significant when the model size increases from T5 large to T5 XL (3B).

Why are there large differences across different settings in Table 1? We discuss this in Section 5.1. The core ideas are pasted here: Our hypothesis is that solving different tasks requires different forms of information. For example, to solve classification tasks with in-context learning, it is crucial to determine the label space and the format by which a classification problem is converted to natural language. In Figure 3, we inspect the behaviors of **SAICL** and FiD on different types of tasks under the *test* domain in the HR→LR setting, where all types of tasks are covered by both training and test domains. One observation is that the benefits of **SAICL** are more pronounced on classification tasks. We hypothesize that this is because classification tasks require more subtle knowledge about the format and label space, which benefits from the fusion of demonstrations enabled by **SAICL** in the encoder. This observation also coincides with the finding in Min et al. (2022b) that in-context learning relies on the label space, input format, and distribution of input text to work.

Can we pre-train SAICL model? We discuss this in Section 6. It is a promising direction to explore using unsupervised objectives to pre-train/fine-tune a **SAICL** model. The key challenge here is constructing training samples in the in-context learning format. A possible approach is to use an additional retrieval module to find similar sentences as demonstrations and use the language modeling objective on the test sample.

Why is the speed advantage of SAICL less significant when there are fewer demonstrations? We discuss this in Section 3. We hypothesize that this is mainly due to the implementation overhead of **SAICL**. During implementation, we mainly focus on the asymptotic complexity of **SAICL** when scaling the number of demonstrations (As we can see in Figure 2). As a result, the implementation might introduce constant overhead compared to well-optimized full attention. We plan to further investigate the overhead by profiling the computation and trying to optimize the computation kernel of **SAICL**.

Why do other works show that the improvement from scaling demonstrations saturates quickly? We discuss this in Section 5.1. Taking Min et al. (2021) as an example, where the authors show that the performance saturates with only 32 demonstrations. We hypothesize that the difference is because the 1024 length limitation of the GPT-2 model used in Min et al. (2021) constrains the effective context, and aggressive truncation is needed here. This finding suggests that the potential of boosting performance with more demonstrations might be underestimated because of inappropriate architectural design.

B. Python-style Pseudo-code of SAICL

```
# We heavily use Einstein style notation for tensor operations
from einops import rearrange
# equivalent to torch.einsum
from opt_einsum import contract


def StructuredAttention(query_states, key_states, value_states, mask, segment_length):
    '''
    query_states, key_states, value_states:
        size: (batch_size, num_heads, num_segment * segment_length, dim)
    mask:
        size: (batch_size, 1, 1, num_segment * segment_length)
    segment_length:
        max length for each demonstration/test input

    '''

    # split into blocks
    # (batch_size, num_heads, num_seg, segment_length, dim)
    query_states = rearrange(
        query_states, 'b h (s t) d -> b h s t d', t=segment_length)
    key_states = rearrange(
        key_states, 'b h (s t) d -> b h s t d', t=segment_length)
    value_states = rearrange(
        value_states, 'b h (s t) d -> b h s t d', t=segment_length)
    # (batch_size, 1, num_seg, 1, key_length)
    block_mask = rearrange(mask, 'b h t (s r) -> b h s t r', r=segment_length)

    # the diagonal part
    # shape (batch_size, num_heads, num_seg, segment_length, segment_length)
    block_diag = contract(
        'b h s t d, b h s r d -> b h s t r', query_states, key_states)

    # calculate position bias for each block
    # this function follows the original T5 relative position bias
    position_bias_diag = calculate_position_bias(
        segment_length, segment_length)
    # (batch_size, num_heads, 1, segment_length, segment_length)
    block_diag_bias = rearrange(position_bias_diag, 'b h t r -> b h 1 t r')
    # add bias to diagonal part
    block_diag += block_diag_bias + block_mask

    # last block column, attention from all demonstrations to the test sample
    block_global_key = contract(
        'b h s t d, b h r d -> b h s t r', query_states[:, :, :-1], key_states[:, :, -1])
    block_global_key += block_mask[:, :, -1, None]
    # last row, attention from the test sample to all demonstrations
    block_global_query = contract(
        'b h t d, b h s r d -> b h s t r', query_states[:, :, -1], key_states[:, :, :-1])
    block_global_query += block_mask[:, :, :-1]

    # merge block diagonal and last column for computing softmax
```

```
        block_global_key_cat = torch.cat(
            [block_diag[:, :, :-1], block_global_key], dim=-1)
        block_global_key_cat = nn.functional.softmax(
            block_global_key_cat.float(), dim=-1).type_as(block_global_key_cat)
        # merge last row for computing softmax
        block_global_query_cat = torch.cat(
            [block_global_query, block_diag[:, :, -1, None]], dim=2)
        block_global_query_cat = rearrange(
            block_global_query_cat, 'b h s t r -> b h t (s r)')
        block_global_query_cat = nn.functional.softmax(
            block_global_query_cat.float(), dim=-1).type_as(block_global_query_cat)

        # dropout
        block_global_key_cat = nn.functional.dropout(
            block_global_key_cat, p=self.dropout, training=self.training
        )
        block_global_query_cat = nn.functional.dropout(
            block_global_query_cat, p=self.dropout, training=self.training
        )

        # split back for computing block matrix multiplication
        output_key_diag = contract('b h s t r, b h s r d -> b h s t d',
                    block_global_key_cat[..., :segment_length], value_states[:, :, :-1])
        output_key_global = contract('b h s t r, b h r d -> b h s t d',
                        block_global_key_cat[..., segment_length:], value_states[:, :, -1])
        output_query_global = contract('b h r l, b h l d -> b h r d', block_global_query_cat,
                        rearrange(value_states, 'b h s t d -> b h (s t) d'))
        output_query_global = rearrange(
            output_query_global, 'b h r d -> b h 1 r d')
        output = torch.cat([output_key_diag+output_key_global,
                            output_query_global], dim=2)

        attn_output = rearrange(
            output, 'b h s t d -> b (s t) (h d)', t=segment_length)

        return attn_output
```

C. Detailed Experimental Settings

Data processing. We follow the MetaICL codebase to process the data. The input in the in-context learning setting consists of several demonstrations and the test sample. We first choose a sample-wise max length (256 for T5 baseline models and $\min(256, \text{max length in the same task})$ for **SAICL** and FiD), then truncate each demonstration $(y_i, \mathbf{x}_i)$ and y_{test} when the length exceeds the max length and pad to max length when using **SAICL** and FiD. We pack as many demonstrations as the number of demonstrations is at most k, and the total length of the input is at most $64k$ as the final input. This way, the final number of demonstrations will range from $k/4$ to k, and for most tasks, it will be $\approx k$.

Training details. We use Adafactor optimizer (Shazeer & Stern, 2018) with learning rate $1e-4$ (following previous works of fine-tuning T5 (Ye et al., 2022b)) and batch size 4. With the linear learning rate schedule, the learning rate first linearly increases from 0 to $1e-4$ for 10% steps and then linearly decreases to 0. The total optimization steps we use is 25600.

D. Additional Experimental Results

We summarize an overview of different methods in Table 2. Specifically, when the prompt contains multi-shot demonstrations, in the encoder, we can choose between sparse attention (i.e., **SAICL**) and full attention. When we need to fuse the information from multiple prompts, we can choose between (1) feeding them independently through the decoder and averaging the logits, and (2) concatenating all the processed tokens and feeding them once to the decoder. We use *Group-FiD* to refer to the case when we construct multiple prompts, each with multiple demonstrations, and concatenate all the encoded tokens as the input to the decoder.

Method	Encoder Attention	Fusion Scheme	Permutation Invariance?	Linear Complexity?
-	Single-shot	Single-prompt	-	-
Baseline	Full	Single-prompt	No	No
SAICL	Sparse	Single-prompt	Yes	Yes
Ensemble $k = 1$	Single-shot	Average	Yes	Yes
Baseline Ensemble	Full	Average	No	No
SAICL Ensemble	Sparse	Average	Yes	Yes
FiD	Single-shot	Concat	Yes	Yes
Baseline + Group-FiD	Full	Concat	No	No
SAICL + Group-FiD	Sparse	Concat	Yes	Yes

Table 2. Comparison of different methods. *Single-shot* means each prompt only contains one demonstration. *Single-prompt* means we only construct one prompt, which may contain multiple demonstrations. *Sparse* means we use **SAICL** in the encoder attention, and *Full* means we use the standard full attention. *Average* means we average the logits of all the predictions from different prompts, and *Concat* means we concatenate all the encoded tokens from different prompts as the input to the decoder.

D.1. Comparison among Single-shot-prompted Methods

We test the performance of ensembling T5 baseline models where each instance only contains 1 demonstration and compare the results with the Fusion-in-Decoder method. For both methods, the k single-shot prompts are processed independently through the encoder. The difference is that, in the fusion-in-decoder method, the processed tokens are concatenated for computing the logit in the decoder, while for the direct ensemble method, the processed tokens are used for computing the logit in the decoder independently and then the logits are averaged.

The results are reported in Figure 6(a). We see that the ensemble of single-shot predictions performs worse than FiD method and saturates when $k \geq 8$. All the results are tested using the corresponding models with train $k = 64$ on HR→LR *unseen domain* setting.

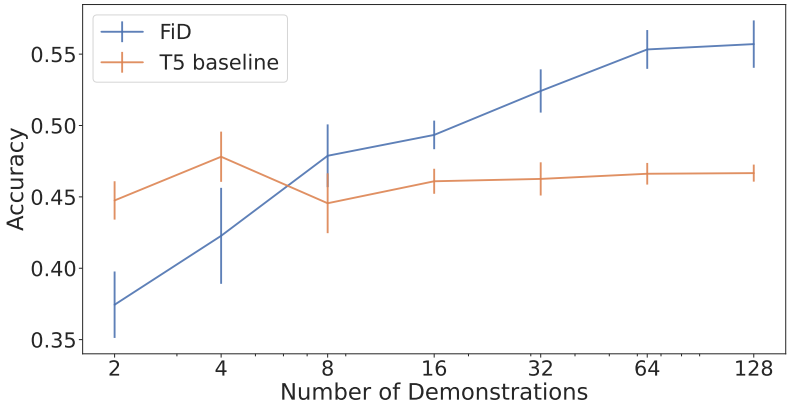
D.2. Comparison between Group-FiD and Ensemble

In the subsection, we test *Group-FiD* method, which resembles the approach proposed in (Hao et al., 2022; Ratner et al., 2022). In Group-FiD, we divide the demonstrations equally among G groups, where each group may possesses multiple demonstrations. We pass these groups independently through the encoder and then concatenate the processed tokens and send them into the decoder. The method can be combined with both **SAICL** and the full attention baseline.

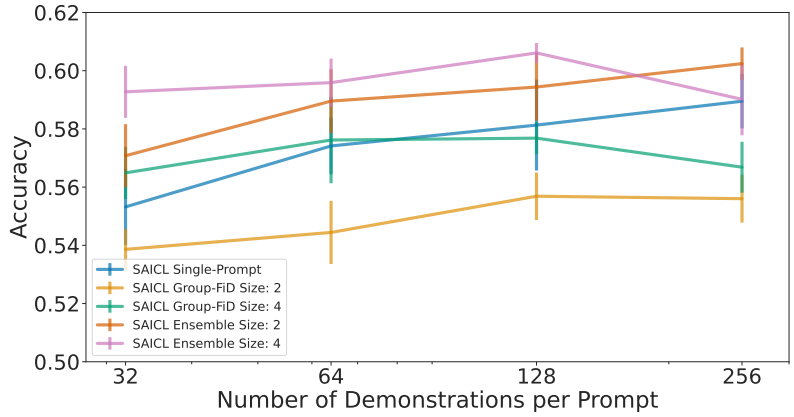
We compare Group-FiD with ensemble and report the results for **SAICL** in Figure 6(b) and T5 baseline in Figure 6(c). We see that for both **SAICL** and the baseline, Group-FiD scheme falls behind the ensemble scheme. All the results are tested using the corresponding models with train $k = 64$ on HR→LR *unseen domain* setting.

E. Other Related Work

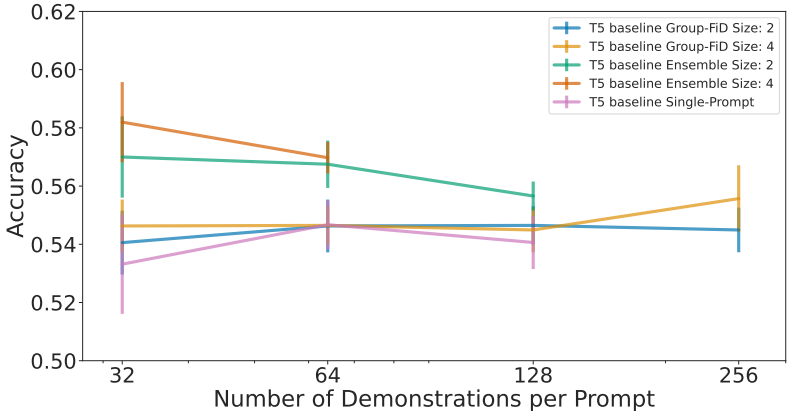
(Few-shot) fine-tuning. The traditional way of adapting a pre-trained language model to new tasks is to fine-tune the model. Recent works on instruction tuning (Sanh et al., 2021; Wei et al., 2021) show that fine-tuning language models on a collection of datasets described via instructions—substantially improves *zero-shot* performance on unseen tasks. Yet, these methods require a diverse set of datasets with a large amount of data, and the training cost is usually high because the whole model has to be updated. Several methods seek to solve the efficiency problem and fine-tune the model in a



(a)



(b)



(c)

Figure 6. **(a)**: Comparison between Ensemble and FiD. We see that direct ensembling the predictions usually perform worse than the FiD method, and saturate quickly when $k \geq 8$. **(b)**: Comparison between Ensemble and Group-FiD for **SAICL**. We see that Ensemble performs better than Group-FiD. **(c)**: Comparison between Ensemble and Group-FiD for T5 baseline. We see that Ensemble performs better than Group-FiD. All the results are tested using the corresponding models with train $k = 64$ on HR→LR *unseen domain* setting.

parameter-efficient way. This is achieved by only tuning the prompt (Lester et al., 2021; Li & Liang, 2021; Gao et al., 2021; Liu et al., 2021; An et al., 2022; Chen et al., 2022b), tuning a subset of the parameters (Zaken et al., 2022; Sung et al., 2021; Guo et al., 2021), or adding a few additional tunable parameters and fixing the original model (Rebuffi et al., 2017; Houlsby et al., 2019; Henderson et al., 2021; Liu et al., 2022a; Bapna & Firat, 2019; Hu et al., 2021). These methods can largely reduce the computational cost of fine-tuning, but they still need at least thousands of updates and thousands of samples. In comparison, in-context learning only involves one forward pass and requires fewer samples. With our methods, the forward complexity only linearly scales to the number of demonstrations, which is more efficient fine-tuning.