Batch-ICL: Effective, Efficient, and Order-Agnostic In-Context Learning

Kaiyi Zhang[1]*, **Ang Lv**[1]*, **Yuhan Chen**[1],
Hansen Ha[2], **Tao Xu**[2], **Rui Yan**[1,3]†
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Ant Group
[3]Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education
{kyzhang02}@gmail.com, {anglv,yuhanchen,ruiyan}@ruc.edu.cn
{hahansen.hhs,tomas.xt}@antgroup.com

Abstract

In this paper, by treating in-context learning (ICL) as a meta-optimization process, we explain why LLMs are sensitive to the order of ICL examples. This understanding leads us to the development of Batch-ICL, an effective, efficient, and order-agnostic inference algorithm for ICL. Differing from the standard N-shot learning approach, Batch-ICL employs N separate 1-shot forward computations and aggregates the resulting meta-gradients. These aggregated meta-gradients are then applied to the forward computation of a zero-shot query to generate the final prediction. This batch processing approach renders the LLM agnostic to the order of ICL examples. Through extensive experiments and analysis, we demonstrate that Batch-ICL consistently outperforms most permutations of ICL examples. In some cases, it even exceeds the performance of the best order for standard ICL, all while reducing the computational resources required. Furthermore, we develop a novel variant of Batch-ICL featuring multiple "epochs" of meta-optimization. This variant implicitly explores permutations of ICL examples, further enhancing ICL performance.[1]

1 Introduction

Brown et al. (2020) demonstrate the capacity of large language models (LLMs) to perform in-context learning (ICL) wherein the input context comprises a handful of illustrative instances of specific tasks. In this few-shot setting, LLMs are capable of identifying the task and adapting their response format and domain accordingly. For instance, when presented with context such as "I love this movie. Sentiment: positive. \n I hate this movie. Sentiment: negative. \n This film is interesting. Sentiment:," the LLM might accurately recognize the sentiment classification task and provide the appropriate response, which in this case would be "positive." Without training or fine-tuning, ICL sometimes even matches the performance of supervised trained models.

Numerous studies (Olsson et al., 2022; Wang et al., 2023b; Dai et al., 2023; Li et al., 2023; Akyürek et al., 2023; Von Oswald et al., 2023; Ren and Liu, 2023; Xie et al., 2022; Lv et al., 2024) have contributed to understanding the mechanism of ICL. Specifically, some research (Dai et al., 2023; Ren and Liu, 2023) describes ICL as a meta-optimization where an LLM is utilized as a meta-optimizer. Meta-gradients are produced through forward processing with ICL examples. These meta-gradients are then applied to the language model through the attention mechanism, resulting in an effective ICL model.

We propose that these insights shed light on a commonly recognized issue: the ICL capacity of an LLM is highly influenced by the order of examples. As emphasized by (Lu et al., 2022), changing the order of ICL examples can lead to significantly different outcomes. This paper offers a preliminary explanation for this phenomenon: In an N-shot ICL process, the meta-gradient is formed based on N sequentially presented examples. Due to the causal attention mechanism in LLMs, the meta-gradient shaped by any given example is indirectly affected by the ones that came before it. Consequently, the order of these examples plays a critical role in shaping the final ICL model. This process is similar to training a neural network with N samples where the batch size is one. In such cases, the gradients generated by each sample are influenced not only by the sample itself but also by the parameter updated by preceding samples. The sample order leads to variations in parameters and the overall performance.

Given that gradients from each sample represents a local optimum and is susceptible to causing

*Equal contribution.
†Corresponding author: Rui Yan (ruiyan@ruc.edu.cn)
[1]Our code is available at https://github.com/Cardinalere/Batch-ICL.

Findings of the Association for Computational Linguistics ACL 2024, pages 10728–10739

suboptimal results, we suggest there is much potential for improving standard ICL. In this paper, we introduce Batch-ICL, an effective, efficient and order-agnostic inference algorithm for in-context learning. Our method diverges from the standard ICL, which typically employs a single N-shot process for an N-shot ICL task. Instead, we implement N separate 1-shot forward computations. This is then followed by the aggregation of N corresponding meta-gradients at a specific layer. These aggregated meta-gradients are then applied to the LLM in the same layer during the forward computation of a zero-shot query, ultimately generating the final predictions. This is equivalent to increase the meta-batch size in ICL from 1 to N, thereby reducing the randomness of meta-optimization and obtaining a better ICL model. Batch-ICL yields three key advantages:

(1) Batch-ICL alleviates concerns related to the order of examples in ICL. Across a variety of tasks, Batch-ICL consistently exhibits improved accuracy compared to the average accuracy achieved through all permutations of ICL examples. It sometimes outperforms the best order. Meanwhile, Batch-ICL reduces the computational resources needed for executing an ICL sequence.

(2) Additionally, we have discovered that although the standard ICL exhibits instability when presented with different ordered sequences, there are advantages to be gained from the interaction among sequential examples. We expand Batch-ICL into a "multi-epoch" variant, which implicitly enumerates the order permutations in a much more efficient manner, leading to further enhancements.

(3) In Batch-ICL, there is no limit to the number of demonstration examples provided that the length of each example does not surpass the pretrained context length. This effectively overcomes a significant constraint encountered in standard N-shot ICL.

2 Understanding In-Context Learning from a Meta-Optimization Perspective

Many works (Aizerman et al., 1964; Irie et al., 2022; Dai et al., 2023; Ren and Liu, 2023) have demonstrated the similarity between the linear attention and the linear layers optimized by gradient descent. In this section, we briefly review this similarity and then introduce the in-context learning from a meta-optimization perspective.

2.1 Dual Form between Linear Attention and Gradient Descent in Linear Layers

Consider a linear layer defined as:

$$f(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x},$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d_{out}\times d_{in}}$ represents the initial weight matrix. Given a sequence of input vectors $\boldsymbol{x}_i \in \mathbb{R}^{d_{in}}, i \in [1, N]$, and their corresponding labels $\boldsymbol{y}_i \in \mathbb{R}^{d_{out}}$, the error signal $\boldsymbol{e}_i$ is produced by backpropagation, where $\boldsymbol{e}_i = -\eta\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}_i}$, with η as the learning rate and $\mathcal{L}$ as the loss function. The weight matrix updates as follows:

$$\boldsymbol{W}' = \boldsymbol{W}_0 + \Delta\boldsymbol{W} = \boldsymbol{W}_0 + \sum_i^N \boldsymbol{e}_i \otimes \boldsymbol{x}_i. \quad (1)$$

Recap that a linear attention is formulated as:

$$\mathtt{LA}(\boldsymbol{V}, \boldsymbol{K}, \boldsymbol{q}) = \boldsymbol{V}^\top\boldsymbol{K}\boldsymbol{q} = \sum_i \boldsymbol{v}_i(\boldsymbol{k}_i^\top\boldsymbol{q}). \quad (2)$$

When focusing on the current input $\boldsymbol{x}_N$, we can derive the dual form between the linear layer optimized by gradient descent and the linear attention:

$$\begin{aligned} f(\boldsymbol{x}_N) &= (\boldsymbol{W}_0 + \Delta\boldsymbol{W})\boldsymbol{x}_N \\ &= \boldsymbol{W}_0\boldsymbol{x}_N + \sum_i^{N-1}(\boldsymbol{e}_i \otimes \boldsymbol{x}_i)\boldsymbol{x}_N \\ &= \boldsymbol{W}_0\boldsymbol{x}_N + \sum_i^{N-1}\boldsymbol{e}_i(\boldsymbol{x}_i^\top\boldsymbol{x}_N) \\ &= \boldsymbol{W}_0\boldsymbol{x}_N + \mathtt{LA}(\boldsymbol{E}, \boldsymbol{X}_{1:N-1}, \boldsymbol{x}_N). \end{aligned} \quad (3)$$

3 Method

3.1 ICL is an Meta-Optimization

In an N-shot ICL setting, a Transformer (Vaswani et al., 2017)-based LLM consists of K layers. The LLM processes an input sequence in the form of $I_1, l_1, I_2, l_2, \ldots, I_N, l_N, I_q$, where I_i and l_i together represent a demonstration example, with I_i being the input and l_i as the corresponding label. Here, I_q denotes the genuine query, and the LLM's task is to predict its label. We represent the embedding of the input sequence at k-layer as $\boldsymbol{X}^k = [\boldsymbol{x}_1^k, \boldsymbol{x}_2^k, \ldots, \boldsymbol{x}_q^k]$ with a special focus on $\boldsymbol{x}_q^k$, which represents the last query token's representation. Without ambiguity, we will omit layer superscripts. The output of an attention head can be expressed as follows:

$$f(\boldsymbol{x}_q) = \boldsymbol{W}_V\boldsymbol{X}\ \mathtt{Softmax}\left(\frac{(\boldsymbol{W}_K\boldsymbol{X})^T\boldsymbol{W}_Q\boldsymbol{x}_q}{\sqrt{d_{out}}}\right), \quad (4)$$

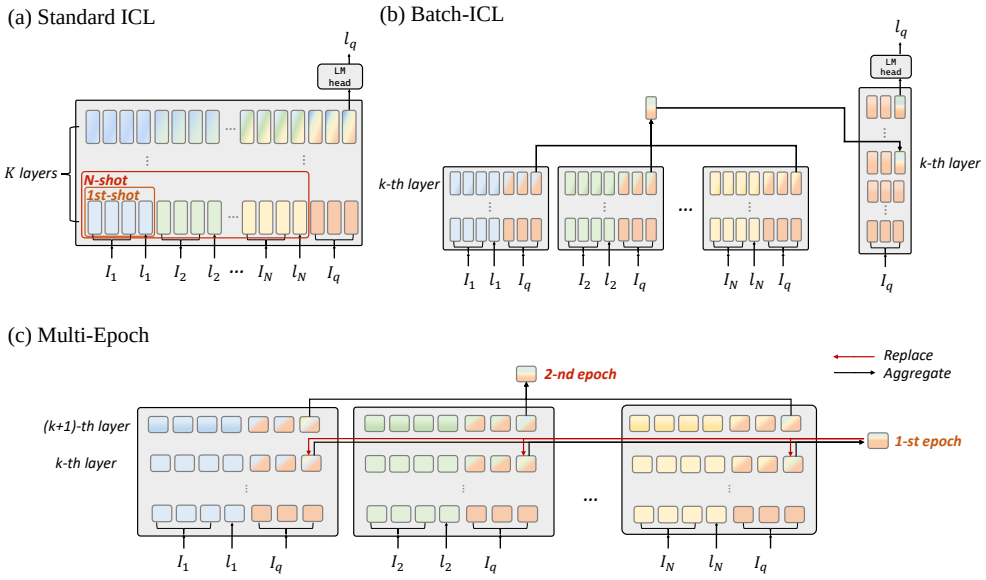


Figure 1: (a) Standard in-context learning. (b) Batch-ICL aggregates the meta-gradients generated during individual 1-shot learning forward computations and applies them to a zero-shot forward process. (c) Multi-epoch Batch-ICL further enhances ICL performance, shown here with a 2-epoch overview.

where $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ are projection matrices belonging to $\mathbb{R}^{d_{out} \times d_{in}}$. These matrices are utilized to compute the attention queries, keys, and values, respectively. To simplify the notation below, we denote $\boldsymbol{W}_Q \boldsymbol{x}_q$ as $\boldsymbol{q}$ and partition $\boldsymbol{X}$ into $[\boldsymbol{X}', \boldsymbol{x}_q]$.

Dai et al. (2023) describe ICL as a meta-optimization process, inspired by the previously mentioned "dual form." In their framework, the LLM acts as the meta-optimizer, with the meta-gradient according to demonstrations generated during the forward computation and applied to the model through the attention mechanism:

$$
\begin{aligned}
f(\boldsymbol{x}_q) &\approx \boldsymbol{W}_V[\boldsymbol{X}'; \boldsymbol{x}_q](\boldsymbol{W}_K[\boldsymbol{X}'; \boldsymbol{x}_q])^\top \boldsymbol{q} \\
&= \underbrace{\boldsymbol{W}_V \boldsymbol{x}_q (\boldsymbol{W}_K \boldsymbol{x}_q)^\top}_{\text{Denoted as } \boldsymbol{W}_0} \boldsymbol{q} + \\
&\quad \underbrace{\mathtt{LA}(\boldsymbol{W}_V \boldsymbol{X}', \boldsymbol{W}_K \boldsymbol{X}', \boldsymbol{q})}_{N \text{ demonstrations' effect, denoted as } \Delta \boldsymbol{W}_N \boldsymbol{q}} \\
&= (\boldsymbol{W}_0 + \Delta \boldsymbol{W}_N)\, \boldsymbol{q}.
\end{aligned} \tag{5}
$$

In the equation above, the standard attention is approximated to a relaxed linear attention. $\boldsymbol{W}_V \boldsymbol{x}_q (\boldsymbol{W}_K \boldsymbol{x}_q)^\top$ is denoted as $\boldsymbol{W}_0$ because $\boldsymbol{W}_0 \boldsymbol{q}$ is the attention output in the 0-shot setting. According to Eq. 3, the outputs from linear attention can be interpreted as the effect of N demonstrations. Consequently, this term is denoted as $\Delta \boldsymbol{W}_N \boldsymbol{q}$.

In this section, we start by proposing an explanation (§3.2) about the sensitivity of LLMs to the order of ICL demonstration examples, which is built upon our comprehension of ICL as a meta-optimization process. Following this, in §3.3, we introduce our proposed solution, named "Batch-ICL," in which the LLM performs an N-shot ICL by processing ICL examples in a "batch." In §3.4, we elaborate on the extension of Batch-ICL to incorporate multiple "epochs," thus fully harnessing its potential.

3.2 The Order of ICL Examples Affects Model Outputs

From this meta-optimization perspective, we provide an explanation for the well-known issue of LLMs' sensitivity to the order of ICL examples (Lu et al., 2022).

Considering the term $\Delta \boldsymbol{W}_N$ in Eq. 5, the causal attention in LLMs permits only later tokens to attend to earlier ones. This implies that altering the order of demonstration examples will correspondingly change the content of $\boldsymbol{X}'$, thereby affecting the meta-gradients $\Delta \boldsymbol{W}_N$ and ultimately the resulting different obtained ICL model and its outputs. We would like to compare this meta-optimization process to optimizing a neural network: for N training samples with a batch size of one, it requires N back-propagation steps. During each step, the gradients are computed and iteratively update the model. The gradients for each training sample depend not only on the sample itself but also on the

parameters updated by preceding samples. Therefore, the order of training examples influences the final model parameters and, consequently, its overall performance.

Despite the significant impact of the orders on performance, this explanation also applies to the observation by Lu et al. (2022) that an increase in the number of examples is correlated with greater variance in performance. This occurs because a larger set of examples raises the upper limit of potential performance. However, due to the meta-gradients being computed sequentially from individual samples, the optimization process is directed towards diverging outcomes, most of which are suboptimal.

3.3 ICL Examples Batching and Meta-Gradients Aggregation

Inspired by the analogy above, we develop an ICL inference algorithm named "Batch-ICL" which empowers LLMs to handle each ICL example and produce the corresponding meta-gradients individually, mirroring the way they are arranged in a training batch. By aggregating these meta-gradients and then applying them to a zero-shot forward process, we have observed that LLMs exhibit superior performance compared to standard ICL.

We adhere to the terminology used in Section 2. Considering a set of N demonstration examples, we opt for the LLM to undertake N individual 1-shot ICL learning processes, diverging from the standard single N-shot learning. Each sample received by LLMs is formatted as "I_i, l_i, I_q", and i spans the entire set N. At a selected layer k, we collect $f_i(\boldsymbol{x_q})$ at the last position, i.e., the result of Eq. 5. Since $\boldsymbol{X}'$ in Eq. 5 now represents the representation of "I_i, l_i", as opposed to the entire set of ICL examples, we accordingly adjust the formula to: $f_i(\boldsymbol{x_q}) = (\boldsymbol{W}_{0,i} + \Delta\boldsymbol{W}_{1,i})\boldsymbol{q}$.

When aggregating these hidden states $[f_i(\boldsymbol{x_q})]_{i=1}^{N}$ using the arithmetic mean, we actually aggregate meta-gradients for each 1-shot learning:

$$\begin{aligned}\bar{f}(\boldsymbol{x_q}) &= \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x_q}) \\ &= \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{W}_{0,i} + \Delta\boldsymbol{W}_{1,i})\boldsymbol{q} \\ &\approx (\boldsymbol{W}_0 + \frac{1}{N}\sum_{i=1}^{N}\Delta\boldsymbol{W}_{1,i})\,\boldsymbol{q}.\end{aligned} \tag{6}$$

The approximation here is due to the minor variations between $\boldsymbol{W}_{0,i}$ and the actual zero-shot weight $\boldsymbol{W}_0$, influenced by x_q being affected by the preceding 1-shot example.

Next, we substitute the zero-shot outputs $\boldsymbol{W}_0\boldsymbol{q}$ at the same layer k with the aggregated $\bar{f}(\boldsymbol{x_q})$, to obtain the final prediction.

To determine the value for parameter k, taking into account the variability introduced by model size, parameters, and tasks, we employ a general approach: For any given task and LLM, we evaluate k across the range of maximum layers. We then choose the k that yields the best performance on the validation set. The selected k remains constant during the testing phase for this LLM and task.

We name this inference algorithm as "Batch-ICL" because it is akin to computing gradients during the optimization of a neural network using a batch of inputs.

3.4 Expanding to Multiple "Epochs"

We notice that the meta-optimization perspective can be extended to encompass "multiple epochs," which in turn further enhances Batch-ICL. To clarify this concept, let's start by distinguishing the layers with superscripts k in Eq. 5:

$$f^{k+1}(\boldsymbol{x_q}^{k+1}) = f^{k+1}(f^{k}(\boldsymbol{x_q}^{k})). \tag{7}$$

Here, we simplify the feed-forward layer by treating it as a linear transformation, considering it as a unit matrix for clarity purposes. When we substitute the aggregated $\bar{f}^k(\boldsymbol{x_q}^k)$ for its individual components $f_i^k(\boldsymbol{x_q}^k)$, which are the separate outputs of each 1-shot ICL at layer k, and then derive the attention outputs from each $k+1$ layer for further aggregation, we actually engage in a form of meta-optimization during an additional epoch. Formally, the outputs of each 1-shot ICL at layer $k+1$ now turns to:

$$\begin{aligned}&f_i^{k+1}(\bar{f}^k(\boldsymbol{x_q}^k)) \\ &= (\boldsymbol{W}_0^{k+1} + \Delta\boldsymbol{W}_i^{k+1})(\boldsymbol{W}_0^k + \frac{1}{N}\sum_{j=1}^{N}\Delta\boldsymbol{W}_j^k)\boldsymbol{q} \\ &= (\boldsymbol{W}_0^{k+1} + \Delta\boldsymbol{W}_i^{k+1})\frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{W}_0^k + \Delta\boldsymbol{W}_j^k)\boldsymbol{q} \\ &= \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{W}_0^{k+1} + \Delta\boldsymbol{W}_i^{k+1})(\boldsymbol{W}_0^k + \Delta\boldsymbol{W}_j^k)\boldsymbol{q}.\end{aligned} \tag{8}$$

Notice the expression $(\boldsymbol{W}_0^{k+1} + \Delta\boldsymbol{W}_i^{k+1})(\boldsymbol{W}_0^k + \Delta\boldsymbol{W}_j^k)\boldsymbol{q}$, which can be understood as conducting

meta-optimization for example j in the first k layers, followed by a similar meta-optimization for example i at layer $k+1$. We term this procedure an extra "epoch" within the Batch-ICL, and it can readily be extended to multi-epoch. Figure 1(c) illustrates the 2-epoch expansion of Batch-ICL.

Essentially, the multi-epoch Batch-ICL implicitly enumerates all permutations of ICL examples, yet it achieves this more efficiently. To understand this, we formulate $f_i^{k+1}({\boldsymbol{x_q}}^{k+1})$ as:

$$\begin{aligned} f_i^{k+1}({\boldsymbol{x_q}}^{k+1}) &= \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{W}_0^{k+1} + \Delta\boldsymbol{W}_{ji}^{k})\boldsymbol{q} \\ &= (\boldsymbol{W}_0^{k+1} + \frac{1}{N}\sum_{j=1}^{N}\Delta\boldsymbol{W}_{ji}^{k})\boldsymbol{q}, \end{aligned} \tag{9}$$

where $\Delta\boldsymbol{W}_{ji}^{k}$ denotes the meta-gradient produced through optimizing example j in the first k layers, followed by optimization of example i in the $k+1$ layer. By aggregating $f_i^{k+1}({\boldsymbol{x_q}}^{k+1})$ across $i = 1 \cdots N$, we derive:

$$\begin{aligned} \bar{f}_i^{k+1}(\boldsymbol{x}_{\boldsymbol{q}}^{k+1}) &= \frac{1}{N}\sum_{i=1}^{N} f_i^{k+1}(\boldsymbol{x}_{\boldsymbol{q}}^{k+1}) \\ &= (\boldsymbol{W}_0^{k+1} + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Delta\boldsymbol{W}_{ji}^{k})\boldsymbol{q}, \end{aligned} \tag{10}$$

In this two-epoch setting, it's evident that N^2 orders of 2-shot examples are contemplated in the summation of meta-gradients. This implicit permutation not only maintains but also, to some extent, amplifies the interaction between examples, thereby enhancing the approach, all the while effectively preserving order-agnostic properties.

3.5 Discussion on the Efficiency

We focus on the computational overhead related to the attention mechanism in Transformer models. This part is a critical efficiency bottleneck for Transformers, showcasing a time complexity of $O(n^2)$ when processing texts of length n.

Consider inputs $(I_1, l_1, I_2, l_2, \ldots, I_N, l_N, I_q)$, and suppose the average length of both an input and a label is T. In a standard N-shot ICL task, the total length of N examples combined with the query is $(2N+1)T$, resulting in an approximate time cost of $O((2N+1)^2T^2)$. In contrast, Batch-ICL deconstructs the N-shot scenario into N separate 1-shot instances, which are subsequently integrated with a 0-shot learning. Specifically, we process (I_i, l_i, I_q), each with a length of $3T$, N times, and (I_q) with a length of T once, culminating in a total time cost of approximately $10T^2$.

When $N > 2$, it is obvious that Batch-ICL outperforms the standard ICL in terms of efficiency. It's important to note that, in practical applications, we execute the N 1-shot learning tasks simultaneously in a batch, which effectively reduces the actual latency. Furthermore, in many situations, particularly in classification tasks, the length of the label is considerably shorter than that of the input. Therefore, the benefits of our approach are even more pronounced.

To analyze practical resource utilization, we tested the AGnews dataset utilizing N=4 on an A100-80G. Batch-ICL completes the inference of the entire dataset in 512.84 seconds, while the standard ICL method requires 697.09 seconds. Considering space efficiency, Batch-ICL provides N times greater space efficiency than standard ICL under ideal conditions. In practice, executing the standard 4-shot AGnews task on Llama 2-13B consumes 32.1 GB of memory, while our method requires only 29.7 GB, wherein model parameters occupy 26 GB.

4 Comparison between Batch-ICL and standard N-shot ICL

4.1 Experimental Setup

We conduct this preliminary study to assess Batch-ICL in comparison to the standard N-shot ICL. We choose widely-used open-source LLMs, Llama-2-7B and -13B (Touvron et al., 2023).

We evaluate Batch-ICL on classification tasks, containing one sentiment detection dataset **SST-2** (Socher et al., 2013), two natural language inference datasets **RTE** (Dagan et al., 2006; Wang et al., 2019) and **QNLI** (Wang et al., 2018), a topic classification dataset **AGNews** (Zhang et al., 2015) and one paraphrase dataset **MRPC** (Dolan and Brockett, 2005). Moreover, we also utilize Batch-ICL for free-format generation tasks, such as machine translation. We choose the **WMT2014 En-Fr** (Bojar et al., 2014) benchmark.

Additionally, we compare Batch-ICL with Parallel Context Windows (PCW, Ratner et al., 2023) that also enhances the inference of ICL. In terms of parallel processed examples, PCW is the most related work to our study. PCW resets the position embedding for each example, implicitly handling these examples in parallel. However, it does not

Task	7B					13B				
	N-shot	Batch-ICL	Best	PCW	F-Ordered	N-shot	Batch-ICL	Best	PCW	F-Ordered
SST-2	54.9 [6.4]	**58.7** [9.2]	58.8 [9.4]	54.2 [6.9]	56.4 [11.8]	63.8 [15.1]	**69.7** [11.4]	70.2 [11.5]	51.9 [3.7]	67.7 [10.9]
RTE	53.0 [6.8]	**65.2** [1.9]	67.6 [0.5]	56.6 [5.2]	53.9 [5.1]	78.1 [4.2]	**79.2** [3.5]	80.9 [2.8]	56.5 [7.3]	80.5 [1.3]
QNLI	50.7 [1.8]	**52.4** [1.3]	55.2 [0.7]	50.1 [0.6]	51.3 [0.8]	51.1 [2.5]	**56.7** [0.9]	56.9 [1.2]	51.2 [1.1]	56.2 [3.9]
AGNews	28.7 [3.8]	29.3 [3.1]	29.4 [3.0]	25.8 [2.0]	**30.2** [4.9]	30.5 [6.3]	31.4 [3.9]	31.6 [3.8]	25.5 [0.8]	**32.1** [6.5]
MRPC	40.5 [10.5]	**66.9** [0.5]	68.0 [0.2]	56.8 [14.3]	38.6 [7.6]	52.4 [14.8]	56.5 [0.6]	67.9 [1.7]	**63.8** [8.0]	63.3 [6.5]

Table 1: Experimental results of classification tasks. We report the average score and standard deviation [σ] across 10 runs. The "Best" column reports the upper limit of Batch-ICL in which we search for the optimal k for each test sample. We also compare Batch-ICL with PCW (Ratner et al., 2023) and Fantastically Ordered (Lu et al., 2022). Batch-ICL not only enhances LLMs' ICL performance in diverse tasks but also decreases performance variability across different demonstration examples, often approaching its maximum potential. All bold results passed the t-test with a p-value < 0.05.

	N-shot	Batch-ICL
7B	66.38	65.91
13B	67.88	67.63

Table 2: BLEU scores for WMT2014 En-Fr.

improve efficiency and introduces gaps in inference behavior, potentially compromising performance. We also compare Batch-ICL with Fantastically Ordered (F-Ordered, Lu et al., 2022), a prior study on selecting the optimal order for ICL. F-Ordered necessitates the enumeration of all permutations to identify an optimal prompt sequence.

For classification tasks, we evaluate the accuracy on the test set when labels are accessible, and we sample 300 data from the validation set to derive the optimal value of k. In cases where test labels are not provided, we evaluate performance on the validation set and choose 300 data from the training set for obtaining k. For the machine translation task, we determine the optimal value of k using the validation set and report the BLEU score (Papineni et al., 2002).

In this section, we fix the value of N as 4. Unless specifically stated otherwise, we present the average score obtained from 10 runs, each with different sampled demonstration examples. This ensures reliable conclusions, as different ICL demonstrations lead to varying performance (Zhao et al., 2021; Perez et al., 2021).

4.2 Experimental Results

We present the experimental results on classification tasks in Table 1. Overall, Batch-ICL demonstrates a substantial improvement over both the standard N-shot ICL and PCW in various tasks and across different model sizes. Across four out of the five datasets analyzed, Batch-ICL surpasses the performance of F-Ordered, with marginal differences observed on the AGnews dataset.

To determine Batch-ICL's maximum potential, we identified the optimal k for each test sample and have included these findings in the "Best" column. These results show that the performance discrepancy between our chosen k and the theoretical upper limit of performance is negligible. Furthermore, because Batch-ICL employs a larger batch size during meta-optimization, which diminishes noise and randomness in the meta-gradients, we observe Batch-ICL exhibits markedly more stable performance than standard ICL across different demonstrations, as evidenced by the significantly reduced variation[2] in repeated experiments. These results serve as initial validation for the effectiveness of Batch-ICL, and also support our explanation in Section 3.2.

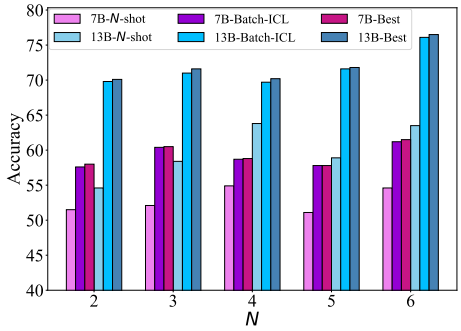
Table 2 illustrates the performance comparison for the machine translation task. Batch-ICL outperforms the standard ICL in efficiency. It achieves this while preserving the BLEU score, experiencing only a minimal decrease of 0.7% for the 7B model and 0.3% for the 13B model, respectively.

To showcase the robustness and versatility of Batch-ICL, we have implemented it across different LLMs, including OPT-6.7B (Zhang et al., 2022) and Falcon-7B (Almazrouei et al., 2023), using SST-2 as a benchmark task. Table 3 illustrates the effectiveness of Batch-ICL across various models, demonstrating its universality.

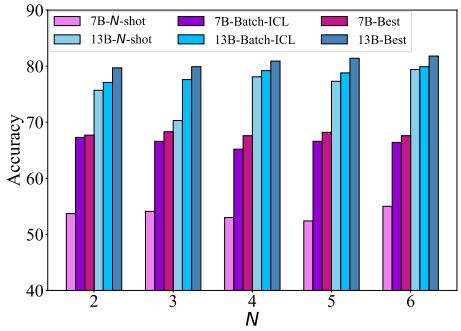
[2]An exception occurs with Llama-2-7B in the SST-2 task, where Batch-ICL increases variance. This happens because, in this binary classification task, the "N-shot" is close to stable random guessing.

	N-shot	Batch-ICL
OPT-6.7B	38.4	42.5
Falcon-7B	51.2	52.0

Table 3: Experimental results of SST-2 on a broader array of LLMs.



(a) SST-2



(b) RTE

Figure 2: Performance dynamics across various N on SST-2 and RTE.

5 Method Analysis

We delve into various aspects of Batch-ICL, including the impact of shot number (N), aggregation layer index (k), the impact of the order of ICL examples, and the optimization "epochs." This exploration aims to provide a deeper insight into both the efficacy of Batch-ICL and the ICL itself.

5.1 The Effect of N

Different tasks typically demand varying values of N. The effectiveness of Batch-ICL across different N represents its robustness in various practical applications. As illustrated in Figure 2, our Batch-ICL consistently surpasses the standard N-shot accuracy for different values of N. Also, the gap between the performance of Batch-ICL and its upper limit referred to as "Best" is minor and diminishes as the value of N increases.

Due to the parallel processing of ICL examples, as long as each individual example doesn't exceed the maximum context length, there's no limit to the number of examples Batch-ICL can handle. In Table 4, for the AGNews task, as N increases from 4 to 70, we found the performance first hits a peak and then drops on Llama-2-7B; By contrast, Batch-ICL keeps bringing benefits with a larger N on Llama-2-13B. We assume the drop in performance at $N = 70$ on Llama-2-7B may stem from the fact that 7B model has already exploited its ability, while 13B is stronger at learning from more shots. Notably, even when we include $N = 70$ examples, averaging a total of 5,447 tokens, which significantly exceeds Llama-2's context limit of 4,096, Batch-ICL continues to demonstrate additional enhancements.

These analyses demonstrate that Batch-ICL is robust to N and can more thoroughly exploits LLMs' in-context learning capacity.

	$N = 4$	$N = 10$	$N = 20$	$N = 70$
7B	29.3	36.24	39.43	33.3
13B	31.4	32.67	33.70	39.1

Table 4: Results from Batch-ICL on the AGNews task with varying values of N. Even when $N = 70$, demonstration examples surpass the model's maximum context length, yet it still achieves additional improvements.

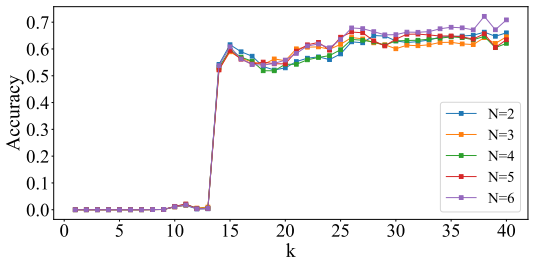


Figure 3: Performance dynamics across various aggregation layer k.

5.2 The Effect of k

We examined the effects of k by varying its value across the full range of layers, monitoring the performance dynamics as aggregated meta-gradients are implemented at various levels. For each value of k, we carried out experiments using the Llama-2-13B model on the SST-2 dataset (Socher et al., 2013), with N values ranging from 2 to 6.

Figure 3 demonstrates that with a small k, Batch-ICL does not work. It suggests that the initial, or shallow, layers of LLMs play a crucial role in establishing semantic foundations for the subsequent,

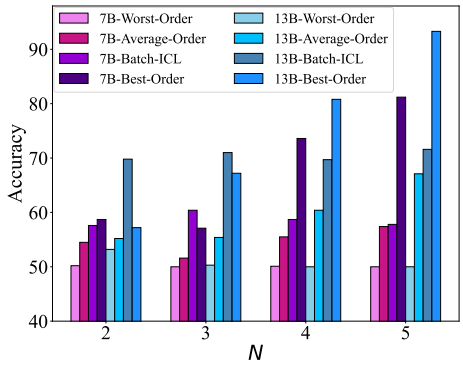


Figure 4: Comparing Batch-ICL and standard ICL with various example orders, including the "Best", "Worst" and "Average" of all permutations.

Model \ N	2	3	4	5
Llama-2-7B	60.0	90.0	65.4	63.7
Llama-2-13B	80.0	88.3	71.3	60.5

Table 5: The percentage of permutations that Batch-ICL outperforms.

deeper layers. This observation aligns with previous studies (Wang et al., 2023b; Todd et al., 2023; Hendel et al., 2023).

As the forward computation goes on, we observe an abrupt increase in performance. This improvement plateaus rapidly and does not diminish thereafter. This pattern differs from the findings reported in (Hendel et al., 2023; Todd et al., 2023), where researchers found that a deeper representation of an ICL task diminishes the task-related information it contains. Specifically, when the representation is extracted and used in a zero-shot forward computation, the accuracy tends to converge to zero as the layer depth increases. The discrepancy between ours and theirs findings might be due to subtle differences in input configuration. In our approach, each 1-shot forward computation uses a true query. In contrast, previous studies used a pseudo query, which can lead to significant deviations in deeper layers because most information gathers at the last position (Wang et al., 2023b).

5.3 The Effect of ICL Example Order

We meticulously examine all $N!$ possible permutations of ICL examples in a N-shot SST-2 (Socher et al., 2013) task. Due to the prohibitively high cost associated with enumerating the permutations, we restrict the value of N within the range of 2 to 5. In Figure 4, the "Average" result is the average accuracy of all permutations. "Best Order" denotes the highest accuracy, contrasting with the "Worst Order" which represents the lowest.

It is clear that, regardless of the model's size, the performance under the worst order is close to a random guess. In stark contrast, the best order significantly surpasses the average performance. This emphasizes the critical role of sequence organization in in-context learning (Lu et al., 2022).

In situations where only a few examples are available (e.g., N=2 or 3), Batch-ICL not only exceeds the average performance but also surpasses the best order. This underscores the strength of our proposed method in data-limited scenarios. As the number of examples increases, the number of permutations rises, enhancing the likelihood of achieving high accuracy; consequently, the best-order performance improves markedly. Nevertheless, Batch-ICL consistently demonstrates superior performance compared to the average.

Additionally, we performed a statistical analysis on the percentage of permutations that Batch-ICL outperforms. In Table 5, it is obvious that Batch-ICL exceeds the majority of permutations for every N. This is particularly noticeable for a smaller N and a larger model. Overall, this study confirms our motivation, demonstrating that Batch-ICL effectively alleviates concerns related to ICL example orders, leading to a satisfactory solution.

5.4 The Effect of Epochs

In Table 6, we present the results of adding more epochs to Batch-ICL on SST-2 (Socher et al., 2013) and RTE (Dagan et al., 2006; Wang et al., 2019). This table illustrates the model's performance across various numbers of N and a range of epochs. Our findings indicate that, across the majority of N and model sizes, there is an improvement in the model's performance over several epochs. Specifically, in the RTE task, we observe a performance plateau typically achieved within 2 epochs. Conversely, in the SST-2 task, we extend our investigation across more epochs, typically reaching a plateau after approximately 10 to 20 epochs, as illustrated in Table 7.

Overall, we observed a more pronounced effect in the 7B model compared to the 13B model. The 7B model consistently shows enhanced performance with an increasing number of epochs, peaking at 4 epochs. On the other hand, the 13B model reaches the plateau earlier. This could be due to the larger capacity of the 13B model, which

N		7B					13B			
	Epoch	**1**	**2**	**3**	**4**	**Epoch**	**1**	**2**	**3**	**4**
2	**SST-2**	57.97	58.78	58.91	**59.48**	**SST-2**	70.09	**70.30**	69.79	69.83
	RTE	67.7	**67.8**	67.7	67.9	**RTE**	79.7	**80.0**	79.5	79.7
3	**SST-2**	60.49	64.27	65.11	**67.77**	**SST-2**	**71.58**	71.35	71.39	71.40
	RTE	68.2	**68.3**	68.3	68.3	**RTE**	79.9	**80.1**	80.0	80.0
4	**SST-2**	58.84	60.52	60.60	**62.10**	**SST-2**	**70.19**	69.90	69.93	69.83
	RTE	67.6	**67.9**	67.8	67.7	**RTE**	80.9	80.9	81.1	**81.3**
5	**SST-2**	57.80	59.96	60.32	**61.39**	**SST-2**	71.80	**71.82**	71.37	71.26
	RTE	68.2	68.3	**68.4**	68.1	**RTE**	81.4	**81.6**	81.4	81.4
6	**SST-2**	61.49	63.08	63.32	**64.57**	**SST-2**	76.52	**76.55**	76.26	75.84
	RTE	67.6	67.7	67.8	**67.9**	**RTE**	81.8	**82.0**	82.0	82.0

Table 6: Multiple epochs prove to enhance accuracy on SST-2 and RTE, irrespective of model size and shots. Note that, in this study, we do not determine k based on the validation set but rather employ the optimal value for each individual sample.

	Epoch=4	Epoch=10	Epoch=20
7B	62.10	67.84	67.40
13B	69.83	76.30	72.38

Table 7: Performance of Batch-ICL with more epochs.

enables it to more effectively capture linguistic subtleties and contextual details with fewer implicit interactions among ICL examples (see Eq.10).

6 Related Works

In addition to PCW we compared in §4, our study is also related to the findings of Hendel et al. (2023) and Todd et al. (2023). These works discovered that in few-shot learning tasks, the representation in deep layers carries the task information. Hendel et al. (2023) focused on identifying and aggregating the outputs of induction heads (Olsson et al., 2022). They emphasized the content analysis of aggregated hidden states. In contrast, we focus on a simpler yet effective method for enhancing the ICL and do not differentiate specific heads. When compared to Hendel et al. (2023), our method demonstrates greater efficacy and surpasses the original few-shot learning performance by using zero-shot learning, an achievement not realized by Hendel et al. (2023). Theoretically, our work offers an explanation for the "training set compression" proposed by Hendel et al. (2023) and elucidates why averaged attention activation conveys ICL task information (Todd et al., 2023).

7 Conclusion

From a meta-optimization perspective, we explain ICL example orders' impact on performance. Building on our insights, we introduce Batch-ICL, an efficient and effective algorithm for ICL inference. Batch-ICL processes ICL examples in batches, aggregating their meta-gradients. Aggregated meta-gradients are then applied to a zero-shot forward computation for final predictions. Due to the batch processing, Batch-ICL is agnostic to the order of ICL examples, surpassing the average performance of all order permutations across various tasks, and supports much more examples. We expand Batch-ICL by developing multi-epoch variants that implicitly enumerate permutations of ICL examples, which fosters better interaction between inputs and further improves our method.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC Grant No. 62122089), Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China. This work was supported by Ant Group Research Fund. Ang Lv is supported by the Outstanding Innovative Talents Cultivation Funded Programs 2023 of Renmin University of China.

Limitations

The theoretical foundation of our work is grounded in various studies (Aizerman et al., 1964; Irie et al., 2022; Dai et al., 2023) which takes the attention in Transformers as a linear attention. Some research (Ren and Liu, 2023) suggests that even without this simplification, the conclusions and insights of these studies remain valid. Nevertheless, for the sake of clarity in presentation and ease of comprehension, we adhere to the linear simplification. Additionally, in the section where we explore multiple "epochs," we simplify the feed-forward layer as a linear transformation. This simplification is widely adopted in many works on interpretability (Olsson et al., 2022; Wang et al., 2023a; Yu et al., 2023; Wang et al., 2023c) due to the considerable challenges associated with analyzing nonlinear MLP.

The potential risks of our study are similar to those of other works involving LLMs, as they can sometimes generate toxic responses.

References

M. A. Aizerman, È. M. Braverman, and L. I. Rozonoèr. 1964. Theoretical foundation of potential functions method in pattern recognition. In *Avtomat. i Telemekh.*, volume 25, pages 917–936.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can GPT learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019, Toronto, Canada. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors.

Kazuki Irie, R'obert Csord'as, and Jürgen Schmidhuber. 2022. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. 2023. Transformers as algorithms: Generalization and stability in in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19565–19594. PMLR.

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

Ang Lv, Yuhan Chen, Kaiyi Zhang, Yulong Wang, Lifeng Liu, Ji-Rong Wen, Jian Xie, and Rui Yan. 2024. Interpreting key mechanisms of factual recall in transformer-based language models.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context

learning and induction heads. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

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

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6383–6402, Toronto, Canada. Association for Computational Linguistics.

Ruifeng Ren and Yong Liu. 2023. In-context learning with transformer is really equivalent to a contrastive learning pattern.

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

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2023. Function vectors in large language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023a. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023c. Label words are anchors: An information flow perspective for understanding in-context learning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.