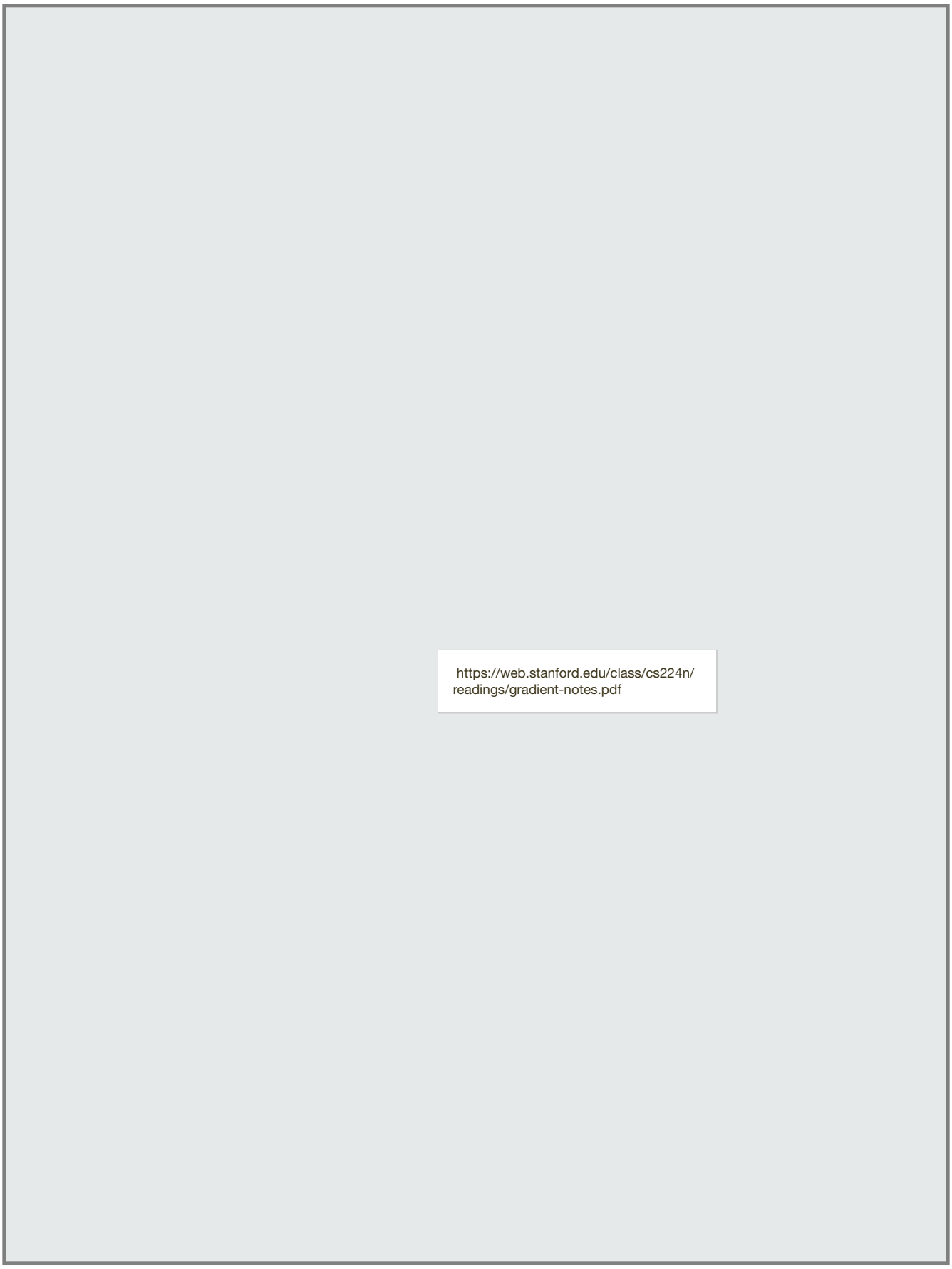

https://web.stanford.edu/class/cs224n/
readings/gradient-notes.pdf

Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta-Optimizers

Damai Dai[†]*, Yutao Sun[‖]*, Li Dong[‡], Yaru Hao[‡], Zhifang Sui[†], Furu Wei[‡]
[†] Peking University [‖] Tsinghua University
[‡] Microsoft Research
https://github.com/microsoft/LMOps

Abstract

Large pretrained language models have shown surprising In-Context Learning (ICL) ability. With a few demonstration input-label pairs, they can predict the label for an unseen input without additional parameter updates. Despite the great success in performance, the working mechanism of ICL still remains an open problem. In order to better understand how ICL works, this paper explains language models as meta-optimizers and understands ICL as a kind of implicit finetuning. Theoretically, we figure out that the Transformer attention has a dual form of gradient descent based optimization. On top of it, we understand ICL as follows: GPT first produces meta-gradients according to the demonstration examples, and then these meta-gradients are applied to the original GPT to build an ICL model. Experimentally, we comprehensively compare the behavior of ICL and explicit finetuning based on real tasks to provide empirical evidence that supports our understanding. The results prove that ICL behaves similarly to explicit finetuning at the prediction level, the representation level, and the attention behavior level. Further, inspired by our understanding of meta-optimization, we design a momentum-based attention by analogy with the momentum-based gradient descent algorithm. Its consistently better performance over vanilla attention supports our understanding again from another aspect, and more importantly, it shows the potential to utilize our understanding for future model designing.

1 Introduction

In recent years, large pretrained language models, especially in Transformer-based architectures (e.g., GPT; Brown et al. 2020), have shown strong emergent In-Context Learning (ICL) ability. Different from finetuning which needs additional parameter updates, ICL just needs several demonstration examples prepended before the original input, and then the model can predict the label for even unseen inputs. On numerous downstream tasks, a large GPT model can achieve a quite great performance, which even exceeds some smaller models with supervised finetuning. However, although ICL has achieved great success in performance, the working mechanism of it is still an open problem to be investigated.

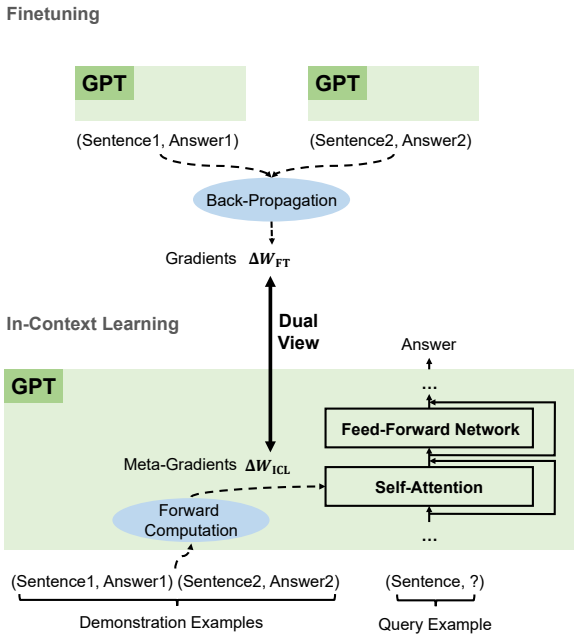


Figure 1: According to the demonstration examples, GPT produces meta-gradients for In-Context Learning (ICL) through forward computation. ICL works by applying these meta-gradients to the model through attention. The meta-optimization process of ICL shares a dual view with finetuning that explicitly updates the model parameters with back-propagated gradients.

In this paper, we explain ICL as a process of meta-optimization and attempt to build connections between GPT-based ICL and finetuning. Concentrating on the attention modules, we figure out that the Transformer attention has a dual form of gradient descent based optimization. On top of it, we

*Contribution during internship at Microsoft Research.

propose a novel perspective to explain ICL: (1) a pretrained GPT serves as a meta-optimizer; (2) it produces meta-gradients according to the demonstration examples through forward computation; (3) the meta-gradients are applied to the original language model through attention to build an ICL model. As illustrated in Figure 1, ICL and explicit finetuning share a dual view of gradient descent based optimization. The only difference is that ICL produces meta-gradients through forward computation, while finetuning computes gradients by back-propagation. Therefore, it is reasonable to understand ICL as some kind of implicit finetuning.

In order to provide empirical evidence to support our understanding, we conduct comprehensive experiments based on real tasks. On six classification tasks, we compare the model predictions, attention outputs, and attention scores of pretrained GPT models in the ICL and finetuning settings. As expected, the behavior of ICL is highly similar to explicit finetuning at all of the prediction level, the representation level, and the attention behavior level. These results are strong evidence to prove the reasonability of our understanding that ICL performs implicit finetuning.

Further, we attempt to take advantage of our understanding of meta-optimization for model designing. To be specific, we design a momentum-based attention, which regards the attention values as meta-gradients and applies the momentum mechanism to them. Experiments on both language modeling and in-context learning show that our momentum-based attention consistently outperforms vanilla attention, which supports our understanding of meta-optimization again from another aspect. We note that beyond this preliminary application, our understanding of meta-optimization may have more potential to be used to aid in model designing, which is worth investigating in the future.

Our contributions are summarized as follows:

- We figure out a dual form between Transformer attention and gradient descent based optimization, and explain language models as meta-optimizers.
- We build connections between ICL and explicit finetuning and propose to understand ICL as a kind of implicit finetuning.
- We provide several lines of empirical evidence to prove that ICL and explicit finetuning behave similarly at multiple levels.
- We design a momentum-based attention that achieves consistent performance improvements, which shows the potential of our understanding of meta-optimization to aid in future model designing.

2 Background

2.1 In-Context Learning with GPT

In this paper, we focus on ICL for classification tasks using GPT (Brown et al., 2020). A GPT model is stacked with L identical Transformer (Vaswani et al., 2017) decoder layers where each layer consists of an attention module and a feed-forward network. For a classification task, given a query input text x and a candidate answer set $Y = \{y_1, y_2, \ldots, y_m\}$, we need to predict a label $\hat{y}$ conditional on n demonstration examples $C = \{(x'_1, y'_1), (x'_2, y'_2), \ldots, (x'_n, y'_n)\}$, where (x'_i, y'_i) is an input-label pair different from the query one. Formally, given a GPT model $\mathcal{M}$, we first compute the probability of each answer y_j:

$$P_{\mathcal{M}}(y_j \mid C, x). \tag{1}$$

Since the label space is restricted for classification, we predict the final answer $\hat{y}$ by selecting the answer with the highest probability from the candidate answer set Y:

$$\hat{y} = y_{\arg\max_j P_{\mathcal{M}}(y_j|C,x)}. \tag{2}$$

In practice, we usually use a pre-defined template to format the demonstrations and prepend them before the query input. Let $\mathcal{T}(\cdot)$ be the function that formats an example, e.g.:

$$\mathcal{T}(x, y) = \text{Sentence: } x. \text{ Sentiment: } y. \tag{3}$$

The contextual model input I is organized like

$$\mathcal{T}(x'_1, y'_1)\ \mathcal{T}(x'_2, y'_2)\ \ldots\ \mathcal{T}(x'_n, y'_n)\ \mathcal{T}(x, \_). \tag{4}$$

Feeding this contextual input into $\mathcal{M}$, the probability of an answer y_j is computed as

$$l_j = \mathcal{M}(I) \cdot \mathbf{e}_{y_j}, \tag{5}$$

$$P_{\mathcal{M}}(y_j \mid C, x) = \text{softmax}(l_j), \tag{6}$$

where $\mathcal{M}(I)$ denotes the output hidden state at the last token position; $\mathbf{e}_{y_j}$ denotes the word embedding of y_j; and l_j is the logit corresponding to the j-th answer.

2.2 Dual Form Between Gradient Descent Based Optimization and Attention

The idea in this paper to explain language models as meta-optimizers is inspired by Aizerman et al. (1964); Irie et al. (2022). They present that linear layers optimized by gradient descent have a dual form of linear attention. Let $W_0, \Delta W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the initialized parameter matrix and the update matrix, respectively, and $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ be the input representation. A linear layer optimized by gradient descent can be formulated as

$$\mathcal{F}(\mathbf{x}) = (W_0 + \Delta W)\,\mathbf{x}. \tag{7}$$

In the back-propagation algorithm, ΔW is computed by accumulating the outer products of the historic input representations $\mathbf{x}_i'^T \in \mathbb{R}^{d_{\text{in}}}$ and the gradients $\mathbf{e}_i \in \mathbb{R}^{d_{\text{out}}}$ of their corresponding outputs:

$$\Delta W = \sum_i \mathbf{e}_i \otimes \mathbf{x}_i'^T. \tag{8}$$

Combing Equation (7) and Equation (8), we can derive the dual form between gradient descent based optimization and linear attention:

$$\begin{aligned}
\mathcal{F}(\mathbf{x}) =& (W_0 + \Delta W)\,\mathbf{x} \\
=& W_0\mathbf{x} + \Delta W\mathbf{x} \\
=& W_0\mathbf{x} + \sum_i \left(\mathbf{e}_i \otimes \mathbf{x}_i'^T\right)\mathbf{x} \\
=& W_0\mathbf{x} + \sum_i \mathbf{e}_i \left(\mathbf{x}_i'^T\mathbf{x}\right) \\
=& W_0\mathbf{x} + \text{LinearAttn}\left(E, X', \mathbf{x}\right),
\end{aligned} \tag{9}$$

where $\text{LinearAttn}(V, K, \mathbf{q})$ denotes the linear attention operation, where we regard the historic output gradients E as values, the historic inputs X' as keys, and the current input $\mathbf{x}$ as the query.

3 In-Context Learning (ICL) Performs Implicit Finetuning

We first qualitatively analyze the Transformer attention under a relaxed linear attention form to figure out a dual form between it and gradient descent based optimization. Then, we compare ICL with explicit finetuning and build connections between these two optimization forms. Based on these theoretical findings, we propose to understand ICL as a kind of implicit finetuning.

3.1 Transformer Attention as Meta-Optimization

Let $\mathbf{x} \in \mathbb{R}^d$ be the input representation of a query token t, and $\mathbf{q} = W_Q\mathbf{x} \in \mathbb{R}^{d'}$ be the attention query vector. In the ICL setting, the attention result of a head is formulated as

$$\begin{aligned}
\mathcal{F}_{\text{ICL}}(\mathbf{q}) =& \text{Attn}(V, K, \mathbf{q}) \\
=& W_V[X'; X]\,\text{softmax}\left(\frac{(W_K[X'; X])^T\,\mathbf{q}}{\sqrt{d}}\right),
\end{aligned} \tag{10}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d' \times d}$ are the projection matrices for computing the attention queries, keys, and values, respectively; $\sqrt{d}$ denotes the scaling factor; X denotes the input representations of query tokens before t; X' denotes the input representations of the demonstration tokens; and $[X'; X]$ denotes the matrix concatenation. For ease of qualitative analysis, we approximate the standard attention to a relaxed linear attention by removing the softmax operation and the scaling factor:

$$\begin{aligned}
\mathcal{F}_{\text{ICL}}(\mathbf{q}) =& \text{Attn}(V, K, \mathbf{q}) \\
\approx& W_V[X'; X]\,(W_K[X'; X])^T\,\mathbf{q} \\
=& W_V X\,(W_K X)^T\,\mathbf{q} + W_V X'\,(W_K X')^T\,\mathbf{q} \\
=& \widetilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}).
\end{aligned} \tag{11}$$

We define $W_{\text{ZSL}} = W_V X\,(W_K X)^T$ as the initialized parameters to be updated since $W_{\text{ZSL}}\mathbf{q}$ is the attention result in the Zero-Shot Learning (ZSL) setting, where no demonstrations are given. Following the reverse direction of Equation (9), we derive a dual form of the Transformer attention:

$$\begin{aligned}
\widetilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}) =& W_{\text{ZSL}}\mathbf{q} + W_V X'\,(W_K X')^T\,\mathbf{q} \\
=& W_{\text{ZSL}}\mathbf{q} + \text{LinearAttn}\left(W_V X', W_K X', \mathbf{q}\right) \\
=& W_{\text{ZSL}}\mathbf{q} + \sum_i W_V\mathbf{x}_i' \left((W_K\mathbf{x}_i')^T\,\mathbf{q}\right) \\
=& W_{\text{ZSL}}\mathbf{q} + \sum_i \left(W_V\mathbf{x}_i' \otimes (W_K\mathbf{x}_i')^T\right)\mathbf{q} \\
=& W_{\text{ZSL}}\mathbf{q} + \Delta W_{\text{ICL}}\mathbf{q} \\
=& (W_{\text{ZSL}} + \Delta W_{\text{ICL}})\,\mathbf{q}.
\end{aligned} \tag{12}$$

As shown in the above equations, the attention to the demonstration tokens is equivalent to parameter updates ΔW_{ICL} that take effect on W_{ZSL}. In addition, By analogy with Equation (9), we can regard $W_V X'$ as some meta-gradients, which are used to compute the update matrix ΔW_{ICL}.

In summary, we explain ICL as a process of meta-optimization: (1) a Transformer-based pretrained language model serves as a meta-optimizer; (2) it produces meta-gradients according to the demonstration examples through forward computation; (3) through attention, the meta-gradients are applied to the original language model to build an ICL model.

3.2 Comparing ICL with Finetuning

In order to compare the meta-optimization of ICL with explicit optimization, we design a specific finetuning setting as a baseline for comparison. Considering that ICL directly takes effect on only the attention keys and values, our finetuning setting also updates only the parameters for the key and value projection. Also in the relaxed linear attention form, the attention result of a finetuned head is formulated as

$$\begin{aligned}\widetilde{\mathcal{F}}_{\text{FT}}(\mathbf{q}) &= (W_V + \Delta W_V)XX^T(W_K + \Delta W_K)^T\mathbf{q} \\ &= (W_{\text{ZSL}} + \Delta W_{\text{FT}})\,\mathbf{q},\end{aligned} \tag{13}$$

where ΔW_K and ΔW_V denote the parameter updates to W_K and W_V, respectively, which are acquired by back-propagation from some training objectives; and ΔW_{FT} is the updates to W_{ZSL} introduced by finetuning.

For a more fair comparison with ICL, we further restrict the finetuning setting as follows: (1) we specify the training examples as the demonstration examples for ICL; (2) we train each example for only one step in the same order as demonstrated for ICL; (3) we format each training example with the same template used for ICL $\mathcal{T}(x'_i, y'_i)$ and use the causal language modeling objective for finetuning.

Comparing ICL and this finetuning setting, we find that ICL has many properties in common with finetuning. We organize these common properties from the following four aspects.

Both Perform Gradient Descent Comparing Equation (12) and Equation (13), we find that both ICL and finetuning introduce updates (ΔW_{ICL} v.s. ΔW_{FT}) to W_{ZSL}, which can both be regarded as gradient descent. The only difference is that ICL produces meta-gradients by forward computation while finetuning acquires real gradients by back-propagation.

Same Training Information The meta-gradients of ICL are produced according to the demonstration examples. The gradients of finetuning are also derived from the same training examples. That is to say, ICL and finetuning share the same source of training information.

Same Causal Order of Training Examples ICL and our finetuning setting share the same causal order of training examples. ICL uses decoder-only Transformers so the subsequent tokens in the demonstrations will not affect the preceding ones. For our finetuning setting, we use the same order of training examples and train only one epoch, so we can also guarantee that the subsequent examples have no effect on the preceding ones.

Both Aim at Attention Compared with zero-shot learning, the direct effect of ICL and our finetuning are both restricted to the computation of attention keys and values. For ICL, the model parameters are unchanged and it encodes demonstration information into additional keys and values to change the attention behavior. For finetuning, due to our restriction, the training information can be introduced to only the projection matrices for attention keys and values as well.

Considering all of these common properties between ICL and finetuning, we think it is reasonable to understand ICL as a kind of implicit finetuning. In the rest of this paper, we compare ICL and finetuning empirically from multiple aspects to provide quantitative results to support this understanding.

Dataset	# Valid. Examples	# Labels
CB	56	3
SST2	872	2
SST5	1101	5
Subj	2000	2
MR	1066	2
AGNews	7600	4

Table 1: Statistics of six classification datasets.

4 Experiments

4.1 Tasks and Datasets

We compare ICL and finetuning based on six datasets spanning three classification tasks. **SST-2** (Socher et al., 2013), **SST-5** (Socher et al., 2013), **MR** (Pang and Lee, 2005) and **Subj** (Pang and Lee, 2004) are four datasets for sentiment classification; **AGNews** (Zhang et al., 2015) is a topic classification dataset; and **CB** (de Marneffe et al., 2019) is used for natural language inference. The statistics of the validation examples and label types are summarized in Table 1.

4.2 Experimental Settings

In our experiments, we use two GPT-like pretrained language models with 1.3B and 2.7B model parameters, respectively, which are released by fairseq[1]. In the rest of this paper, we call them GPT 1.3B and

[1] https://github.com/facebookresearch/fairseq

GPT 2.7B for short. All experiments are conducted on NVIDIA V100 GPUs with 32 GB memory.

For each task, we use the same template to format examples for Zero-Shot Learning (ZSL), ICL, and finetuning. Details of the templates used for each task are provided in Appendix A. The answer prediction processes for ZSL and finetuning are the same with ICL as described in Section 2.1, except that they do not have demonstration examples.

For ICL, we fix the number of demonstration examples to 32 and tune the random seed for each task to find a set of demonstration examples that achieves the best validation performance. For finetuning, we use the same demonstration examples for ICL as the training examples and use SGD as the optimizer. For a fair comparison, we fine-tune the model for only one epoch and the training examples are provided in the same order as demonstrated for ICL. We tune the learning rate for finetuning and select the one that achieves the best validation performance. Details of the search range and selected value for the random seeds and learning rates are shown in Appendix B.

4.3 Metrics

We design three metrics to measure the similarity between ICL and finetuning at three different levels: the prediction level, the representation level, and the attention behavior level.

Recall to Finetuning Predictions (Rec2FTP) At the prediction level, this metric measures ICL can cover how much behavior of finetuning. We first count N_{FT}, the number of query examples that finetuning can predict correctly but ZSL cannot. Then, among these examples, we count N_{both}, the number that ICL can also predict correctly. Finally, we compute the Rec2FTP score as $\frac{N_{\text{both}}}{N_{\text{FT}}}$. A higher Rec2FTP score suggests that ICL covers more behavior of finetuning at the prediction level.

Similarity of Attention Output Updates (SimAOU) This metric measures the similarity between the effects that ICL and finetuning have on ZSL at the representation level. For a query example, let $\mathbf{h}_{\text{X}}^{(l)}$ denote the output representation of the last token at the l-th attention layer in the X setting. The updates of ICL and finetuning compared with ZSL are $\mathbf{h}_{\text{ICL}}^{(l)} - \mathbf{h}_{\text{ZSL}}^{(l)}$ and $\mathbf{h}_{\text{FT}}^{(l)} - \mathbf{h}_{\text{ZSL}}^{(l)}$, respectively. We compute the cosine between these two updates to get the SimAOU score at the l-th layer. A higher SimAOU score means ICL is more inclined to update the attention output representation in the same direction as finetuning.

Model	Task	ZSL	FT	ICL
GPT 1.3B	CB	37.5	57.1	57.1
	SST2	70.5	73.5	92.7
	SST5	39.3	42.0	45.0
	Subj	72.6	78.8	90.0
	MR	65.9	68.2	89.0
	AGNews	46.3	53.7	79.2
GPT 2.7B	CB	42.9	60.7	55.4
	SST2	71.4	91.2	95.0
	SST5	35.9	46.6	46.5
	Subj	75.2	85.6	90.3
	MR	60.9	78.8	91.3
	AGNews	39.8	66.6	80.3

Table 2: Validation accuracy in the ZSL, finetuning, and ICL settings on six classification datasets.

Similarity of Attention Map (SimAM) This metric measures the similarity of the attention behavior of ICL and finetuning. For a query example, let $\mathbf{m}_{\text{X}}^{(l,h)}$ denote the attention weights before softmax of the last token at the h-th attention head in the l-th attention layer in the X setting. For ICL, we omit the attention to the demonstration tokens and only monitor the attention weights to the query input. We compute the cosine between $\mathbf{m}_{\text{ICL}}^{(l,h)}$ and $\mathbf{m}_{\text{FT}}^{(l,h)}$ and then average the similarity across the attention heads to get the SimAM score at each layer. A higher SimAM score indicates that the attention weights that ICL and finetuning pay to the query tokens are more similar.

4.4 Results

Accuracy We first show the validation accuracy in the ZSL, ICL, and finetuning settings on six classification datasets in Table 2. Compared with ZSL, ICL and finetuning both achieve considerable improvements, which means the optimizations they make are both helpful to these downstream tasks. In addition, we find that ICL is better at few-shot scenarios than finetuning.

Rec2FTP We show the Rec2FTP scores for two GPT models on six datasets in Table 3. As shown in the table, on average, ICL can correctly predict 87.64% of the examples that finetuning can correct from ZSL. These results indicate that at the prediction level, ICL can cover most of the correct

Model	Task	Rec2FTP	SimAOU	Random SimAOU	SimAM	ZSL SimAM
GPT 1.3B	CB	91.67	0.189	0.004	0.386	0.152
	SST2	86.32	0.128	0.003	0.608	0.555
	SST5	70.16	0.173	0.004	0.430	0.391
	Subj	84.39	0.070	0.004	0.504	0.378
	MR	92.14	0.188	0.003	0.513	0.398
	AGNews	85.41	0.155	0.003	0.536	0.152
GPT 2.7B	CB	100.00	0.184	-0.001	0.362	0.228
	SST2	93.87	0.113	0.003	0.687	0.687
	SST5	74.32	0.142	0.001	0.411	0.380
	Subj	90.46	0.100	0.004	0.375	0.346
	MR	95.44	0.120	0.001	0.346	0.314
	AGNews	87.48	0.210	-0.003	0.305	0.172

Table 3: Rec2FTP, SimAOU, and SimAM scores on six classification datasets. The demonstrated SimAOU and SimAM scores are averaged across examples and layers. For comparison, we also show two baseline metrics for SimAOU and SimAM, respectively. On all of these datasets, ICL tends to perform similar behavior to finetuning at the prediction, representation, and attention behavior levels.

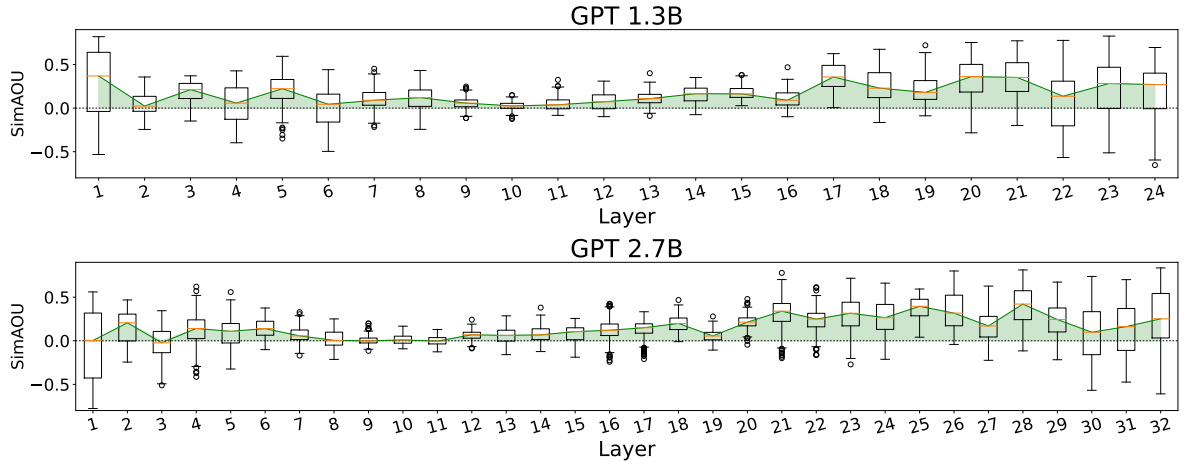


Figure 2: Statistics of the SimAOU scores at different layers. The yellow lines denote medians.

behavior of finetuning.

SimAOU We present the SimAOU scores averaged across examples and layers for two GPT models on six datasets in Table 3. For comparison, we also provide a baseline metric (***Random SimAOU***) that computes the similarity between ICL updates and randomly generated updates. From the table, we find that ICL updates are much more similar to finetuning updates than to random updates, which means at the representation level, ICL tends to change the attention results in the same direction as finetuning changes.

SimAM Table 3 also demonstrates the SimAM scores averaged across examples and layers for two GPT models on six datasets. As a baseline metric for SimAM, ***ZSL SimAM*** computes the similarity between ICL attention weights and ZSL attention weights. Comparing these two metrics, we also observe that compared with ZSL, ICL is more inclined to generate attention weights similar to those of finetuning. Again, at the attention behavior level, we prove that ICL behaves similarly to finetuning.

5 Discussions

5.1 Similarity at Different Layers

In order to investigate the similarity between ICL and finetuning more thoroughly, we look into the SimAOU and SimAM scores at different layers. We randomly sample 50 validation examples from each dataset and draw box plots for SimAOU and

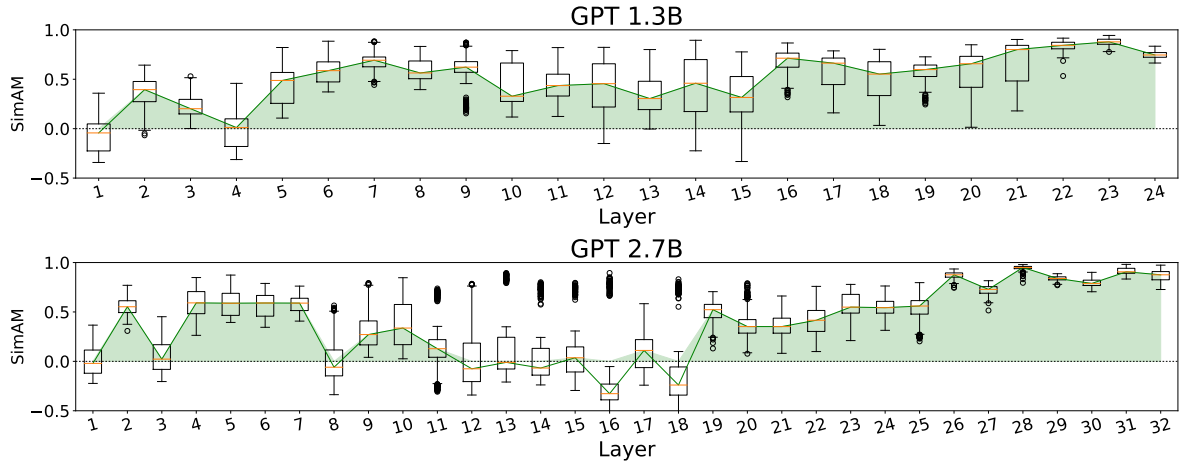
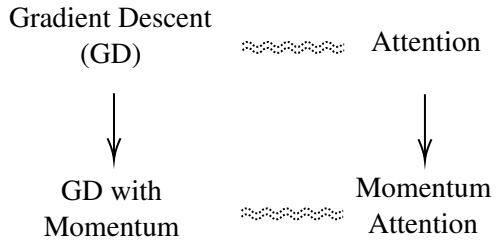


Figure 3: Statistics of the SimAM scores at different layers. The yellow lines denote medians.

SimAM in Figure 2 and Figure 3, respectively. From the figures, we find that both SimAOU and SimAM are fluctuated at lower layers, and tend to be steadily larger at higher layers. This phenomenon suggests that the meta-optimization made by ICL has forward accumulated effects, so with more accumulation, ICL will behave more similarly to finetuning at higher layers.

5.2 Mapping Between Optimization Algorithm and Transformer Architecture



We have figured out the dual form between Transformer attention and gradient descent based optimization. Inspired by this dual view, we investigate whether we can utilize momentum (Polyak, 1964; Sutskever et al., 2013), a widely used technique for improving optimization algorithms, to improve Transformer attention.

Method As stated in Section 3.1, the attention values serve as some kind of meta-gradients. By analogy with momentum SGD that averages gradients among timestamps, we try to apply Exponential Moving Average (EMA) (Hunter, 1986) to the attention values to build momentum-based attention:

$$\begin{aligned}\mathrm{MoAttn}(V, K, \mathbf{q}_t) &= \mathrm{Attn}(V, K, \mathbf{q}_t) + \mathrm{EMA}(V) \\ &= V\,\mathrm{softmax}(\frac{K^T \mathbf{q}_t}{\sqrt{d}}) + \sum_{i=1}^{t-1} \eta^{t-i} \mathbf{v}_i,\end{aligned}$$

where η is a hyper-parameter, and $\mathbf{v}_i$ is the i-th attention value vector. We assume that introducing momentum into attention will help capture long dependency and thus lead to faster convergence and better performance.

Experiments on Language Modeling First, we evaluate the effect of momentum-based attention on language modeling. We train two GPT models with 350M parameters from scratch, where one is the vanilla Transformer, and another applies momentum to attention. We evaluate the perplexity of these two models on the training set and three validation sets with input lengths of 256, 512, and 1024, respectively. The results are shown in Table 4. We find that on all of the validation sets, applying momentum to attention introduces a consistent perplexity improvement compared with the vanilla Transformer.

Experiments on In-Context Learning We also evaluate the in-context learning ability of the above language models to verify the effectiveness of the momentum-based attention on downstream tasks. We consider six datasets for sentiment analysis (SST5 (Socher et al., 2013), IMDB (Maas et al., 2011), MR (Pang and Lee, 2005)), natural language inference (CB (de Marneffe et al., 2019)), and multi-choice selection (ARC-E (Clark et al., 2018), PIQA (Bisk et al., 2020)). For all these datasets, we use 32 examples as demonstrations. As shown in Table 5, compared with the vanilla Transformer, using momentum-based attention achieves consistently higher accuracy in all the tasks.

The performance improvements on both language modeling and in-context learning prove that

As seen from experiments, this momentum based attention improves perplexity (hence performance) ⇒ Meta-optimization hypothesis "proof"

Model	Train_{1024}	Valid_{256}	Valid_{512}	Valid_{1024}
Transformer	17.61	19.50	16.87	15.14
$\text{Transformer}_{\text{MoAttn}}$	**17.55**	**19.37**	**16.73**	**15.02**

Table 4: Perplexity on the training set and validation sets with different input lengths for language modeling. Applying momentum to attention introduces a consistent perplexity improvement compared with the vanilla Transformer.

Model	SST5	IMDB	MR	CB	ARC-E	PIQA	Average
Transformer	25.3	64.0	61.2	43.9	48.2	68.7	51.9
$\text{Transformer}_{\text{MoAttn}}$	**27.4**	**70.3**	**64.8**	**46.8**	**50.0**	**69.0**	**54.7**

Table 5: Accuracy on six in-context learning tasks. Introducing momentum into attention improves the accuracy of the vanilla Transformer by 2.8 on average.

introducing momentum into attention is an effective strategy, which supports our understanding of meta-optimization from another aspect.

6 Conclusion

In this paper, we aim to explain the working mechanism of GPT-based ICL. Theoretically, we figure out a dual form of ICL and propose to understand ICL as a process of meta-optimization. Further, we build connections between ICL and a specific finetuning setting and find that it is reasonable to regard ICL as a kind of implicit finetuning. Empirically, in order to support our understanding that ICL performs implicit finetuning, we comprehensively compare the behavior of ICL and finetuning based on real tasks. The results prove that ICL behaves similarly to explicit finetuning at the prediction level, the representation level, and the attention behavior level. Further, inspired by our understanding of meta-optimization, we design a momentum-based attention that achieves consistent performance improvements. We believe that our understanding will have more potential to aid in ICL application and model designing in the future.

References

Mark A Aizerman, Emmanuil M Braverman, and Lev I Rozonoer. 1964. Theoretical foundation of potential functions method in pattern recognition. *Avtomatika i Telemekhanika*, 25(6):917–936.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *CoRR*, abs/2211.15661.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The CommitmentBank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2022. What can transformers learn in-context? A case study of simple function classes. *CoRR*, abs/2208.01066.

J Stuart Hunter. 1986. The exponentially weighted moving average. *Journal of quality technology*, 18(4):203–210.

Kazuki Irie, Róbert Csordás, 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, ICML 2022*, volume 162 of *Proceedings of Machine Learning Research*, pages 9639–9659. PMLR.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

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.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008. Curran Associates, Inc.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*.

A Templates for In-Context Learning

We demonstrate the templates used to format examples and the candidate answer sets for six classification datasets used in our experiments in Table 6.

B Hyper-parameters

We perform grid search to find the best random seed for ICL and the best learning rate for finetuning. The search range for all the datasets is the same. For random seeds, we search in $\{1, 2, 3, 4, 5, 6, 7\}$. For learning rates, the search base values are $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and we scale them to 0.1, 0.01, and 0.001 times, i.e., we have $9 \times 3 = 27$ values to search.

In Table 7, we present the details of the selected random seeds and learning rates for two GPT models on six classification datasets.

Dataset	Template	Candidate Answer Set
CB	{Premise} Question: {Hypothesis} True, False, or Neither? Answer: {Label}	{ True, False, Neither }
SST-2	Sentence: {Sentence} Label: {Label}	{ Negative, Positive }
SST-5	Sentence: {Sentence} Label: {Label}	{ terrible, bad, neutral, good, great }
Subj	Input: {Sentence} Type: {Label}	{ objective, subjective }
MR	Review: {Sentence} Sentiment: {Label}	{ Negative, Positive }
AGNews	Classify the news articles into the categories of World, Sports, Business, and Technology. News: {Sentence} Type: {Label}	{ World, Sports, Business, Technology }

Table 6: Formatting templates and candidate answer sets for six classification datasets.

Hyper-parameter	Dataset	GPT 1.3B	GPT 2.7B
Random Seed	CB	3	3
	SST2	2	7
	SST5	5	5
	Subj	4	4
	MR	5	1
	AGNews	3	3
Learning Rate	CB	0.100	0.090
	SST2	0.020	0.007
	SST5	0.006	0.003
	Subj	0.003	0.002
	MR	0.010	0.001
	AGNews	0.200	0.060

Table 7: Selected random seeds and learning rates for two GPT models on six classification datasets.