WHY CAN GPT LEARN IN-CONTEXT? LANGUAGE MODELS IMPLICITLY PERFORM GRADIENT DESCENT AS META-OPTIMIZERS

Damai Dai[†,*], Yutao Sun[‖,*], Li Dong[‡], Yaru Hao[‡], Shuming Ma[‡], Zhifang Sui[†], Furu Wei[‡]
[†] Peking University [‖] Tsinghua University [‡] Microsoft Research
{daidamai,szf}@pku.edu.cn
{lidong1,fuwei}@microsoft.com
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 labels for unseen inputs without parameter updates. Despite the great success in performance, its working mechanism still remains an open question. In this paper, we explain language models as meta-optimizers and understand ICL as implicit finetuning. Theoretically, we figure out that Transformer attention has a dual form of gradient descent. 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. We compare the behaviors of ICL and explicit finetuning on real tasks to provide empirical evidence that supports our understanding. Experimental results show that in-context learning behaves similarly to explicit finetuning from multiple perspectives.

1 INTRODUCTION

In this paper, we explain in-context learning (ICL) as a process of meta-optimization and analyze connections between GPT-based in-context learning and finetuning. Concentrating on the attention modules, we figure out that the Transformer attention has a dual form of gradient descent. On top of it, we propose a novel perspective to explain ICL: (1) 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. Figure 1 shows that in-context learning and explicit finetuning share a dual view of gradient descent, where ICL produces meta-gradients while finetuning computes back-propagated gradients. Therefore, we understand in-context learning as implicit finetuning.

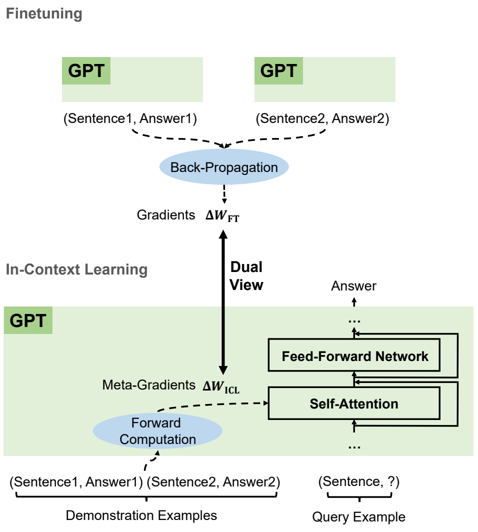

Figure 1: Through forward computation, GPT produces meta-gradients for ICL, which shares a dual view with fine-tuning that updates model parameters with back-propagated gradients.

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, attention weights to query tokens, and attention weights to training tokens between in-context learning and finetuning. Experimental results vali-

[*] Contribution during internship at Microsoft Research.

date that the behavior of in-context learning is similar to explicit finetuning from multiple perspectives. These results are strong evidence to prove the reasonability of our understanding of in-context learning as implicit finetuning.

2 Understanding In-Context Learning as Implicit Finetuning

Inspired by Irie et al. (2022); Aizerman et al. (1964), we first qualitatively analyze the Transformer attention under a relaxed linear attention form to figure out a dual form between it and gradient descent. Then, we compare in-context learning with explicit finetuning to analyze connections between these two optimization forms. Based on these theoretical findings, we propose to understand in-context learning as implicit finetuning.

2.1 Understanding 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

$$\mathcal{F}_{\text{ICL}}(\mathbf{q}) = \text{Attn}(V, K, \mathbf{q}) = W_V[X'; X]\,\text{softmax}\left(\frac{\left(W_K[X'; X]\right)^T \mathbf{q}}{\sqrt{d}}\right), \tag{1}$$

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 relaxed linear attention by removing the softmax operation and the scaling factor:

$$\mathcal{F}_{\text{ICL}}(\mathbf{q}) \approx W_V[X'; X]\left(W_K[X'; X]\right)^T \mathbf{q} = W_V X\left(W_K X\right)^T \mathbf{q} + W_V X'\left(W_K X'\right)^T \mathbf{q} = \widetilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}). \tag{2}$$

We define $W_{\text{ZSL}} = W_V X\left(W_K X\right)^T$ 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 (13) in Appendix A.2, we derive a dual form of the Transformer attention:

$$\begin{aligned}
\widetilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}) &= W_{\text{ZSL}}\mathbf{q} + W_V X'\left(W_K X'\right)^T \mathbf{q} = W_{\text{ZSL}}\mathbf{q} + \text{LinearAttn}\left(W_V X', W_K X', \mathbf{q}\right) \\
&= W_{\text{ZSL}}\mathbf{q} + \sum_i W_V \mathbf{x}'_i\left(\left(W_K \mathbf{x}'_i\right)^T \mathbf{q}\right) = W_{\text{ZSL}}\mathbf{q} + \sum_i \left(\left(W_V \mathbf{x}'_i\right) \otimes \left(W_K \mathbf{x}'_i\right)\right)\mathbf{q} \\
&= W_{\text{ZSL}}\mathbf{q} + \Delta W_{\text{ICL}}\mathbf{q} = \left(W_{\text{ZSL}} + \Delta W_{\text{ICL}}\right)\mathbf{q}.
\end{aligned} \tag{3}$$

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 E in Equation (13), we regard $W_V X'$ as meta-gradients, which are used to compute the update matrix ΔW_{ICL}.

In summary, we explain in-context learning as a process of meta-optimization: (1) a pretrained GPT 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.

2.2 Comparing ICL with Finetuning

Based on the above understanding of in-context learning, we further compare the meta-optimization of in-context learning with the explicit optimization of finetuning to analyze connections between them. Considering that ICL directly takes effect on only the attention keys and values, we design a specific finetuning setting as the compared baseline, which 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

$$\widetilde{\mathcal{F}}_{\text{FT}}(\mathbf{q}) = (W_V + \Delta W_V)XX^T(W_K + \Delta W_K)^T\mathbf{q} = \left(W_{\text{ZSL}} + \Delta W_{\text{FT}}\right)\mathbf{q}, \tag{4}$$

where ΔW_K and ΔW_V denote the parameter updates to W_K and W_V, respectively, which are acquired by back-propagation from task-specific training objectives; and ΔW_{FT} is the updates to

Model	SST2	SST5	MR	Subj	AGNews	CB	Average
GPT 1.3B	91.84	66.67	97.08	87.17	83.08	87.50	85.56
GPT 2.7B	96.83	71.60	95.83	87.63	84.44	100.00	89.39

Table 1: Rec2FTP on six datasets. ICL can cover most of correct predictions of finetuning.

W_{ZSL} introduced by finetuning. For a more fair comparison with in-context learning, we further restrict the finetuning setting as follows: (1) we specify the training examples as the demonstration examples for in-context learning; (2) we train each example for only one step in the same order as demonstrated for in-context learning; (3) we format each training example with the same template used for ICL and use the causal language modeling objective for finetuning.

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

Both Perform Gradient Descent Comparing Equation (3) and Equation (4), we find that both in-context learning and finetuning introduce updates (ΔW_{ICL} v.s. ΔW_{FT}) to W_{ZSL}, which drive from implicit and explicit gradient descent, respectively. The main 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, in-context learning and finetuning share the same source of training information.

Same Causal Order of Training Examples In-context learning 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 in-context learning 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 these common properties, we understand ICL as implicit finetuning. Next, we compare ICL and finetuning empirically to provide quantitative evidence for our understanding.

3 Experiments

We analyze two off-the-shelf pretrained GPT models with 1.3 billion and 2.7 billion model parameters, respectively, which are released by fairseq. In the rest of this paper, we call them GPT 1.3B and GPT 2.7B for short. All experiments are conducted on NVIDIA V100 GPUs with 32 GB memory. Details of the experimental settings, evaluation datasets, and validation accuracy in the zero-shot learning (ZSL), finetuning, and in-context learning (ICL) settings on these datasets in Appendix B

3.1 ICL Covers Most of Correct Predictions of Finetuning

We compute a **recall to finetuning prediction (Rec2FTP)** to measure ICL can cover how much behavior of finetuning from the perspective of the model prediction. We first count $N_{\text{FT>ZSL}}$, the number of query examples that finetuning can predict correctly but ZSL cannot. Then, among these examples, we count $N_{(\text{FT>ZSL})\wedge(\text{ICL>ZSL})}$, the number that ICL can also predict correctly. Finally, we compute the Rec2FTP score as $\frac{N_{(\text{FT>ZSL})\wedge(\text{ICL>ZSL})}}{N_{\text{FT>ZSL}}}$. A higher Rec2FTP score suggests that ICL covers more correct behavior of finetuning from the perspective of the model prediction.

We show the Rec2FTP scores in Table 1. As shown in the table, on average, ICL can correctly predict more than 85% of the examples that finetuning can correct from ZSL. These results indicate that from the perspective of model prediction, ICL can cover most of the correct behavior of finetuning.

Model	Metric	SST2	SST5	MR	Subj	AGNews	CB	Average
GPT 1.3B	SimAOU (Random Δ)	0.002	0.003	0.001	0.002	0.002	0.003	0.002
	SimAOU (ΔFT)	**0.110**	**0.080**	**0.222**	**0.191**	**0.281**	**0.234**	**0.186**
GPT 2.7B	SimAOU (Random Δ)	0.000	-0.002	0.000	0.001	-0.002	0.000	-0.001
	SimAOU (ΔFT)	**0.195**	**0.323**	**0.157**	**0.212**	**0.333**	**0.130**	**0.225**

Table 2: SimAOU on six datasets. From the perspective of representation, ICL tends to change attention output representations in the same direction as finetuning changes.

Model	Metric	SST2	SST5	MR	Subj	AGNews	CB	Average
GPT 1.3B	SimAM (Before Finetuning)	0.555	0.391	0.398	0.378	0.152	0.152	0.338
	SimAM (After Finetuning)	**0.585**	**0.404**	**0.498**	**0.490**	**0.496**	**0.177**	**0.442**
GPT 2.7B	SimAM (Before Finetuning)	**0.687**	0.380	0.314	0.346	0.172	**0.228**	0.355
	SimAM (After Finetuning)	**0.687**	**0.492**	**0.347**	**0.374**	**0.485**	0.217	**0.434**

Table 3: SimAM on six datasets. From the perspective of attention behavior, ICL is more inclined to generate similar attention weights to those after finetuning.

3.2 ICL Changes Attention Outputs in the Same Direction as Finetuning

From the perspective of representation, we compute a **similarity of the attention output updates (SimAOU)** to measure the similarity between the updates that ICL and finetuning make. For a query example, let $\mathbf{h}_{\text{X}}^{(l)}$ denote the normalized output representation of the last token at the l-th attention layer in setting X. 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 **SimAOU (ΔFT)** at the l-th layer. A higher SimAOU (ΔFT) means ICL is more inclined to update attention outputs in the same direction as finetuning. For comparison, we also compute a **SimAOU (Random Δ)** that measures the similarity between ICL updates and randomly generated updates.

We present the SimAOU scores averaged across examples and layers in Table 2. From the table, we find that SimAOU (Random Δ) is always around zero, while SimAOU (ΔFT) remains much more positive. These results indicate that ICL updates are much more similar to finetuning updates than to random updates. From the perspective of representation, we prove that ICL tends to change the attention outputs in the same direction as finetuning.

3.3 ICL Is Inclined to Generate Similar Attention Weights to Finetuning

From the perspective of attention behavior, we compute a **similarity of the attention map (SimAM)** to measure the similarity of the attention map to query tokens for 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 setting X. For ICL, we omit the attention to the demonstration tokens and only monitor the attention weights to the query tokens. First, before finetuning, we compute the cosine between $\mathbf{m}_{\text{ICL}}^{(l,h)}$ and $\mathbf{m}_{\text{ZSL}}^{(l,h)}$ and then average the similarity across attention heads to get **SimAM (Before Finetuning)** at each layer. Similarly, after finetuning, we compute the cosine between $\mathbf{m}_{\text{ICL}}^{(l,h)}$ and $\mathbf{m}_{\text{FT}}^{(l,h)}$ to get **SimAM (After Finetuning)**. A higher SimAM (After Finetuning) over SimAM (Before Finetuning) indicates that the attention behavior of ICL is more similar to a finetuned model than a non-finetuned one.

Table 3 demonstrates the SimAM scores averaged across examples and layers. We observe that compared with attention weights before finetuning, ICL is more inclined to generate similar attention weights to attention weights after finetuning. Again, from the perspective of attention behavior, we prove that ICL behaves similarly to fine-tuning.

Model	Metric	SST2	SST5	MR	Subj	AGNews	CB	Average
GPT 1.3B	Kendall (ICL, Random)	0.000	-0.001	0.000	0.001	-0.001	0.000	0.000
	Kendall (ICL, FT)	**0.192**	**0.151**	**0.173**	**0.181**	**0.190**	**0.274**	**0.193**
GPT 2.7B	Kendall (ICL, Random)	-0.001	0.000	0.000	0.000	0.000	-0.001	0.000
	Kendall (ICL, FT)	**0.213**	**0.177**	**0.264**	**0.203**	**0.201**	**0.225**	**0.214**

Table 4: Kendall rank correlation coefficients for two GPT models on six datasets. Compared with random attention weights, ICL attention weights to training tokens are much more similar to finetuning attention weights.

3.4 ICL and Finetuning Tend to Pay Similar Attention to Training Tokens

Since we understand ICL as a process of meta-optimization, we also compare the attention to training tokens for ICL and finetuning with the **Kendall rank correlation coefficient** (Kendall, 1948). For a query example, let $\mathbf{m}_{\text{ICL}}^{(l)}$ denote the ICL attention weights to the demonstration tokens of the last query token in the l-th attention layer, which is summed across attention heads. For finetuning, we first record all the attention queries $Q'^{(l,h)} \in \mathbb{R}^{d' \times N}$ of the training tokens, and then use the inner product between them and the attention query $\mathbf{q}^{(l,h)} \in \mathbb{R}^{d'}$ of the last token in the query example as the finetuning attention weights to the training tokens: $\mathbf{m}_{\text{FT}}^{(l)} = \sum_h Q'^{(l,h)^T} \mathbf{q}^{(l,h)}$, which is also summed across attention heads. The Kendall coefficient between $\mathbf{m}_{\text{ICL}}^{(l)}$ and $\mathbf{m}_{\text{FT}}^{(l)}$ is computed as **Kendall (ICL, FT)** $= \frac{P_c - P_d}{N(N-1)/2}$, where N denotes the number of training tokens, P_c denotes the number of concordant pairs, and P_d denotes the number of discordant pairs. A higher Kendall coefficient means that the orders of attention weights to training tokens of ICL and finetuning are more similar. For comparison, we also compute the Kendall coefficient between $\mathbf{m}_{\text{ICL}}^{(l)}$ and randomly generated attention weights $\mathbf{m}_{\text{Random}}^{(l)}$, which we call **Kendall (ICL, Random)**.

Table 4 shows the Kendall correlation coefficients averaged across examples and layers for two GPT models on six datasets. We find that Kendall (ICL, Random) is always near zero, while Kendall (ICL, FT) always maintains a distinctly positive value. These results suggest that ICL and finetuning tend to pay similar attention to training tokens.

In addition, in Appendix C, inspired by our dual form, we design a momentum-based attention that achieves consistent performance improvements over vanilla attention, which further supports our understanding of Transformer attention from another perspective.

4 Conclusion

In this paper, we aim to explain the working mechanism of GPT-based ICL. Theoretically, we figure out a dual form between Transformer attention and gradient descent, and propose to understand ICL as a process of meta-optimization. Further, we analyze connections between ICL and explicit finetuning and show the reasonability to regard ICL as implicit finetuning. Empirically, we comprehensively compare ICL and finetuning based on six real NLP tasks. The results prove that ICL behaves similarly to explicit finetuning from multiple perspectives. Further, inspired by our understanding of meta-optimization, we design a momentum-based attention that achieves consistent performance improvements over vanilla attention. We believe our understanding will have more potential to enlighten ICL applications and model design in the future.

References

Mark A Aizerman, Emmanuil M Braverman, and Lev I Rozonoer. Theoretical foundation of potential functions method in pattern recognition. *Avtomatika i Telemekhanika*, 25(6):917–936, 1964. URL `https://www.mathnet.ru/links/87a8bbda73212482651476ffcb127472/at11677.pdf`.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pp. 7432–7439. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

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. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pp. 107–124, 2019. URL https://ojs.ub.uni-konstanz.de/sub/index.php/sub/article/download/601/456/.

J Stuart Hunter. The exponentially weighted moving average. *Journal of quality technology*, 18(4): 203–210, 1986. URL http://jssm.uludag.edu.tr/~orbak/L11-OnEWMA.pdf.

Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. 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*, pp. 9639–9659. PMLR, 2022. URL https://proceedings.mlr.press/v162/irie22a.html.

Maurice George Kendall. Rank correlation methods. 1948.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Bo Pang and Lillian Lee. 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)*, pp. 271–278, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1218990. URL https://aclanthology.org/P04-1035.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer (eds.), *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pp. 115–124. The Association for Computer Linguistics, 2005. doi: 10.3115/1219840.1219855. URL https://aclanthology.org/P05-1015/.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964. URL https://vsokolov.org/courses/750/files/polyak64.pdf.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1139–1147. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/sutskever13.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pp. 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Appendix

A Background

A.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:

$$P_{\mathcal{M}}(y_j \mid C, x). \tag{5}$$

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:

$$\hat{y} = \arg\max_{y_j} P_{\mathcal{M}}(y_j \mid C, x). \tag{6}$$

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.:

$$\mathcal{T}(x, y) = \text{Sentence: } x\text{. Sentiment: } y. \tag{7}$$

The contextual model input I is organized like

$$\mathcal{T}(x'_1, y'_1)\ \mathcal{T}(x'_2, y'_2)\ \ldots\ \mathcal{T}(x'_n, y'_n)\ \mathcal{T}(x, \_). \tag{8}$$

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

$$l_j = \mathcal{M}(I) \cdot \mathbf{e}_{y_j}, \tag{9}$$

$$P_{\mathcal{M}}(y_j \mid C, x) = \operatorname{softmax}(l_j), \tag{10}$$

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

A.2 Dual Form Between Attention and Linear Layers Optimized by Gradient Descent

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

$$\mathcal{F}(\mathbf{x}) = (W_0 + \Delta W)\,\mathbf{x}. \tag{11}$$

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

$$\Delta W = \sum_i \mathbf{e}_i \otimes \mathbf{x}'_i, \tag{12}$$

where $\mathbf{e}_i$ is derived from the historic output gradients by multiplying $-\gamma$, the negative learning rate. Combing Equation (11) and Equation (12), we can derive the dual form of linear layers optimized by gradient descent:

$$\begin{aligned}
\mathcal{F}(\mathbf{x}) &= (W_0 + \Delta W)\,\mathbf{x} \\
&= W_0\mathbf{x} + \Delta W\mathbf{x} \\
&= W_0\mathbf{x} + \sum_i \left(\mathbf{e}_i \otimes \mathbf{x}'_i\right)\mathbf{x} \\
&= W_0\mathbf{x} + \sum_i \mathbf{e}_i \left(\mathbf{x}_i'^T \mathbf{x}\right) \\
&= W_0\mathbf{x} + \operatorname{LinearAttn}\left(E, X', \mathbf{x}\right),
\end{aligned} \tag{13}$$

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

B Experimental Settings

We analyze two off-the-shelf pretrained GPT models with 1.3 billion and 2.7 billion model parameters, respectively, which are released by fairseq.[1] 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), finetuning (FT), and in-context learning (ICL). Details of the templates used for each task are provided in Table 5. The answer prediction processes for ZSL and finetuning are the same as ICL, except that they do not have demonstration examples.

Dataset	Template	Candidate Answer Set

SST2	Sentence: {Sentence}
	Label: {Label}
		{ Negative, Positive }

SST5	Sentence: {Sentence}
	Label: {Label}
		{ terrible, bad, neutral, good, great }

MR	Review: {Sentence}
	Sentiment: {Label}
		{ Negative, Positive }

Subj	Input: {Sentence}
	Type: {Label}
		{ objective, subjective }

AGNews	Classify the news articles into the categories of World, Sports, Business, and Technology.
	News: {Sentence}
	Type: {Label}
		{ World, Sports, Business, Technology }

CB	{Premise}
	Question: {Hypothesis} True, False, or Neither?
	Answer: {Label}
		{ True, False, Neither }

Table 5: Formatting templates and candidate answer sets for six classification datasets.

For in-context learning, we fix the max 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 explicit finetuning, we use the same demonstration examples for in-context learning as the training examples and use SGD as the optimizer. For a fair comparison, we finetune the model for only one epoch and the training examples are provided in the same order as demonstrated for in-context learning. 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.1.

For reference, we present the statistics of six evaluation datasets and validation accuracy in the ZSL, finetuning, and ICL settings on these datasets in Appendix B.2.

B.1 Hyper-Parameters for In-Context Learning and Finetuning

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, 0.001, and 0.0001 times, i.e., we have $9 \times 4 = 36$ values to search. As an exception, for GPT 1.3B finetuned on SST5, we perform a more fine-grained search and finally set its learning rate to 0.00016 since the finetuned model cannot outperform the zero-shot learning with the above 36 learning rates.

In Table 6, we present the details of the selected random seeds and learning rates for two GPT models on six classification datasets.

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

Hyper-Parameter	**Dataset**	**GPT 1.3B**	**GPT 2.7B**
Random Seed	SST2	2	7
	SST5	5	5
	MR	5	1
	Subj	4	4
	AGNews	3	3
	CB	3	3
Learning Rate	SST2	0.0005	0.007
	SST5	0.00016	0.04
	MR	0.003	0.001
	Subj	0.003	0.002
	AGNews	0.2	0.2
	CB	0.08	0.01

Table 6: Selected random seeds and learning rates for two GPT models on six classification datasets.

B.2 Evaluation Datasets and Validation Accuracy

We compare in-context learning and finetuning based on six datasets spanning three sorts of classification tasks. **SST2** (Socher et al., 2013), **SST5** (Socher et al., 2013), **MR** (Pang & Lee, 2005) and **Subj** (Pang & 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. Statistics of the number of validation examples and label types are summarized in Table 7.

For reference, we present the validation accuracy in the ZSL, finetuning, and ICL settings on six classification datasets in Table 7. Compared with ZSL, ICL and finetuning both achieve considerable improvements, which means the optimizations they make are both helpful to these downstream tasks.

	SST2	**SST5**	**MR**	**Subj**	**AGNews**	**CB**
# Validation Examples	872	1101	1066	2000	7600	56
# Label Types	2	5	2	2	4	3
ZSL Accuracy (GPT 1.3B)	70.5	39.3	65.9	72.6	46.3	37.5
FT Accuracy (GPT 1.3B)	73.9	39.5	73.0	77.8	65.3	55.4
ICL Accuracy (GPT 1.3B)	92.7	45.0	89.0	90.0	79.2	57.1
ZSL Accuracy (GPT 2.7B)	71.4	35.9	60.9	75.2	39.8	42.9
FT Accuracy (GPT 2.7B)	76.9	39.1	80.0	86.1	65.7	57.1
ICL Accuracy (GPT 2.7B)	95.0	46.5	91.3	90.3	80.3	55.4

Table 7: Statistics of six classification datasets (rows 1-2) and validation accuracy in the zero-shot learning (ZSL), finetuning (FT), and in-context learning (ICL) settings on these datasets (rows 3-8).

C Momentum-Based Attention Inspired by Dual Form of Transformer Attention

We have figured out the dual form between Transformer attention and gradient descent. As illustrated in Figure 2, inspired by this dual view, we investigate whether we can utilize momentum (Polyak, 1964; Sutskever et al., 2013), a widely used technique for optimization algorithms, to improve Transformer attention.

Gradient descent with momentum averages gradients among timestamps:

$$\Theta_t = \Theta_{t-1} - \gamma \sum_{i=1}^{t-1} \eta^{t-i} \nabla f_{\Theta_i}, \tag{14}$$

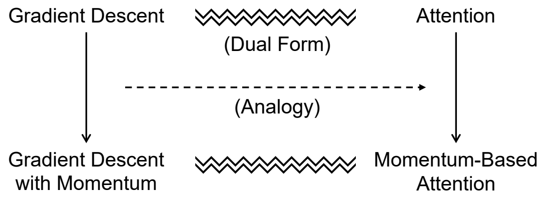

Figure 2: Inspired by the dual form between attention and gradient descent, we introduce the momentum mechanism into Transformer attention by analogy with gradient descent with momentum.

where γ is the learning rate and η is a scalar between 0 and 1. As stated in Section 2.1, the attention values serve as meta-gradients. By analogy with gradient descent with momentum, we try to use Exponential Moving Average (EMA; Hunter 1986) to average the attention values to build the momentum-based attention:

$$\mathrm{MoAttn}(V, K, \mathbf{q}_t) = \mathrm{Attn}(V, K, \mathbf{q}_t) + \mathrm{EMA}(V) = V\,\mathrm{softmax}(\frac{K^T\mathbf{q}_t}{\sqrt{d}}) + \sum_{i=1}^{t-1} \eta^{t-i}\mathbf{v}_i, \tag{15}$$

where $\mathbf{v}_i$ is the i-th attention value vector. The momentum of attention value vectors explicitly strengthens the recency bias of attention, which has been shown helpful for language modeling (Press et al., 2022). Therefore, we assume that introducing momentum into attention will contribute 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. Hyper-parameters for training these two models are provided in Table 8. 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 9. On all of the validation sets, applying momentum to attention introduces a consistent perplexity improvement compared with the vanilla Transformer.

Hyper-parameter	**Value**
Embedding & Hidden Dimension	1024
FFN Inner Hidden Dimension	4096
Number of Attention Heads	16
Number of Transformer Layers	24
Number of Parameters	350M
Sequence Length	1024
Batch Size	512K Tokens
Optimizer	Adam
Adam Betas	(0.9, 0.98)
Adam Epsilon	1e-6
Maximum Learning Rate	3e-4
Learning Rate Scheduler	Polynomial Decay
Total Training Steps	500K
Warm-up Steps	20K
Gradient Clip Norm	2.0

Table 8: Hyper-parameters for training two language models from scratch.

Experiments on In-Context Learning We also evaluate the in-context learning ability of the above language models to verify the effectiveness of momentum-based attention on downstream tasks. We consider six datasets for sentiment analysis (SST5 (Socher et al., 2013), IMDB (Maas et al., 2011), and MR (Pang & Lee, 2005)), natural language inference (CB (De Marneffe et al., 2019)), and multi-choice selection (ARC-E (Clark et al., 2018) and PIQA (Bisk et al., 2020)). For all of these datasets, we use up to 32 examples as demonstrations. As shown in Table 10, compared with vanilla Transformer, using momentum-based attention achieves consistently higher accuracy on all of these datasets.

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

Table 9: Perplexity on the training set and validation sets with different input lengths for language modeling. Momentum-based attention achieves 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
Transformer$_{\text{MoAttn}}$	**27.4**	**70.3**	**64.8**	**46.8**	**50.0**	**69.0**	**54.7**

Table 10: Accuracy on six in-context learning datasets. Introducing momentum into attention improves the accuracy of the vanilla Transformer by 2.8 on average.

The performance improvements on both language modeling and in-context learning prove our deduction that introducing momentum will improve Transformer attention. From another perspective, these results further support our understanding of Transformer attention as meta-optimization.