In-context Learning and Gradient Descent Revisited

Anonymous ACL submission

Abstract

In-context learning (ICL) has shown impressive results in few-shot learning tasks, yet its underlying mechanism is still not fully understood. A recent line of work, suggesting that ICL performs gradient descent (GD)-based optimization implicitly, has captured the imagination of researchers. While promising, the formal results mainly focus on simplified linear settings and provide only a preliminary evaluation of realistic scenarios. In this work, we revisit evidence for ICL-GD correspondence. We identify concerning evidence against these claims. We find gaps in evaluation and theory. We identify flaws in metrics and baselines and insufficient theoretical justification. We show that information flows in ICL and GD in very different ways (we dub this discrepancy *Layer Causality*), and show that a variant of GD aligns with ICL consistently better than vanilla GD. We find that even simple untrained transformers are on par with trained models on ICL-GD correspondence, leading to serious doubts. Our code implementation is available at: https://anonymized.

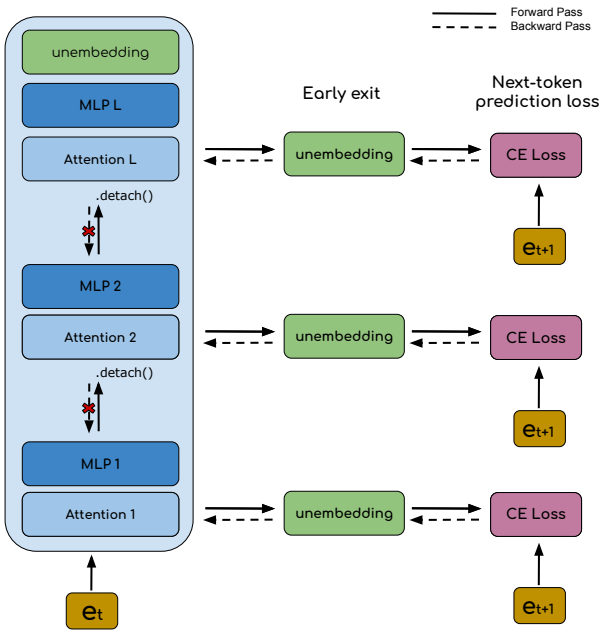

Figure 1: **Layer Causal GD**: The output of each layer is projected to the label space and used as an intermediate prediction. We compute the prediction loss of each intermediate layer sequentially.

1 Introduction

In recent years, large language models have shown strong emergent in-context learning ability (Brown et al., 2020; Wei et al., 2022), where a pretrained model's performance significantly improves on various downstream tasks by simply conditioning on a few input-label pairs (demonstrations). Despite its success and substantial research, the inner workings behind ICL remain elusive. In-context learning operates in a seemingly different manner from finetuning. Nevertheless, a series of recent works discusses apparent similarities between ICL and gradient descent-based optimization (Irie et al., 2022; Von Oswald et al., 2023; Akyürek et al., 2023; Dai et al., 2023). Specifically, there seems to be some mechanism that allows ICL to implement implicit GD using in-context demonstrations as training examples. In this paper, we take issue with arguments made in the literature, showing significant evidence *against* them (at least in the realistic, non-linear case). Most crucially, we find that untrained models perform as good as pretrained models in terms of relevant similarity metrics between ICL and GD.

In terms of theory, we identify a core discrepancy between the two procedures: *Layer Causality* – an intrinsic difference in information flow between ICL and GD. In ICL, the information that influences the hidden state comes from the output of *lower layers alone*. In GD, however, the update to the weights of a layer depends on gradients – coming from deeper layers.

We adopt the framework proposed by (Dai et al.,

	SST2	SST5	MR	Subj	AGNews	CB	Average
$\text{SimAOU}_{\text{norm}}$ (GD)	$0.11_{\pm 0.00}$	$0.09_{\pm 0.01}$	$0.22_{\pm 0.01}$	$0.18_{\pm 0.02}$	$0.31_{\pm 0.04}$	$0.21_{\pm 0.01}$	0.187
$\text{SimAOU}_{\text{norm}}$ (LCGD)	$\mathbf{0.22}_{\pm 0.01}$	$\mathbf{0.11}_{\pm 0.00}$	$\mathbf{0.33}_{\pm 0.01}$	$\mathbf{0.35}_{\pm 0.01}$	$\mathbf{0.33}_{\pm 0.01}$	$\mathbf{0.34}_{\pm 0.00}$	**0.279**
SimAOU (GD)	$0.05_{\pm 0.01}$	$0.04_{\pm 0.02}$	$0.17_{\pm 0.03}$	$0.06_{\pm 0.01}$	$\mathbf{0.18}_{\pm 0.03}$	$0.11_{\pm 0.01}$	0.102
SimAOU (LCGD)	$\mathbf{0.13}_{\pm 0.01}$	$\mathbf{0.11}_{\pm 0.01}$	$\mathbf{0.21}_{\pm 0.03}$	$\mathbf{0.18}_{\pm 0.00}$	$0.13_{\pm 0.01}$	$\mathbf{0.24}_{\pm 0.01}$	**0.167**
SimAM (GD)	$\mathbf{0.59}_{\pm 0.01}$	$\mathbf{0.40}_{\pm 0.03}$	$\mathbf{0.49}_{\pm 0.01}$	$\mathbf{0.45}_{\pm 0.06}$	$\mathbf{0.48}_{\pm 0.04}$	$\mathbf{0.20}_{\pm 0.03}$	**0.435**
SimAM (LCGD)	$0.58_{\pm 0.01}$	$0.39_{\pm 0.03}$	$0.30_{\pm 0.00}$	$0.27_{\pm 0.01}$	$0.12_{\pm 0.00}$	$0.04_{\pm 0.01}$	0.283
SimAM_{Δ} (GD)	$0.15_{\pm 0.02}$	$0.31_{\pm 0.02}$	$0.14_{\pm 0.05}$	$0.25_{\pm 0.07}$	$\mathbf{0.50}_{\pm 0.05}$	$0.25_{\pm 0.01}$	0.267
SimAM_{Δ} (LCGD)	$\mathbf{0.30}_{\pm 0.02}$	$\mathbf{0.33}_{\pm 0.01}$	$\mathbf{0.26}_{\pm 0.00}$	$\mathbf{0.32}_{\pm 0.01}$	$0.43_{\pm 0.02}$	$\mathbf{0.38}_{\pm 0.01}$	**0.336**

Table 1: SimAOU and SimAM comparison of vanilla GD and layer causal GD across six classification datasets. Layer causal GD achieves higher SimAOU across all tasks, yet its SimAM is significantly lower. SimAM_{Δ} is higher for layer causal GD, except for AGNews.

2023) as it allows evaluating natural language tasks on pretrained transformers. In their original work, they demonstrate empirically a better-than-random correspondence between ICL and vanilla GD fine-tuning on GPT models over six classification tasks. We discuss the metrics used, SimAOU and SimAM, and find elusive problems with both of them. We make minor modifications that eliminate those problems. However, we find these modifications significantly diminish similarity scores. We also find problems with some of the datasets used in their benchmark.

Another weakness with current works is the lack of fair comparison. In (Dai et al., 2023), which we discuss extensively due to its realistic setting, they compare only to a baseline of random vectors, which gets a score of almost zero. We propose two new benchmarks: (1) a straightforward modification that takes into account the layer causal limitations of ICL, (2) untrained models (with and without trained embeddings). We show that layer causal GD is superior to vanilla GD in terms of similarity scores. We show that even untrained models are *on par with pretrained models* – despite not having learned to perform ICL yet, or even a good representation for tokens (in the case of untrained embeddings). This suggests that most of the similarity comes from shallow features such as text repetition. This means that tailored weight constructions as in (Von Oswald et al., 2023) and (Akyürek et al., 2023) are not relevant to non-linear transformers.

Our contributions are the following:

- We highlight core problems with the hypothesis that GD approximates ICL.
- We present an alternative to GD and empirically demonstrate that it is better at simulating ICL.
- We survey existing literature and tackle potential gaps in evaluation, in terms of baseline methods, evaluation metrics, and datasets.
- When considering the above modifications, we find it sensible to argue against ICL-GD correspondence in realistic non-linear transformers. We leave it for future research to investigate more closely the extent to which some of these conclusions apply to linear transformers.

2 Background and Preliminaries

2.1 Notation

We use uppercase $A, B, C, X, Y, Z, ...$ to denote matrices and bold lowercase letters $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{x}, \mathbf{y}, \mathbf{z}, ...$ for vectors. We define attention as:

$$\text{Attn}(V, K, \mathbf{q}) = V \,\text{softmax}\left(\frac{K^T \mathbf{q}}{\sqrt{d}}\right)$$

where $K, V \in \mathbb{R}^{d \times d}, \mathbf{q} \in \mathbb{R}^d$.

2.2 Linear Attention and Gradient Descent

The view of language models as meta-optimizers originates from the presentation of the dual and primal forms of the perceptron (Aizerman et al., 2019). This notion was later expressed in terms of key-value-query attention operation by (Irie et al., 2022; Dai et al., 2023; Von Oswald et al., 2023). They show that linear layers optimized by GD have a dual representation as linear attention. We

use the linear attention model, whereby the softmax operation and the scaling factor are omitted: $V \text{softmax}(\frac{K^T \mathbf{q}}{\sqrt{d}}) \approx VK^T q =: \text{LA}(V, K, \mathbf{q})$. In linear attention, $W = VK^T$ can be thought of as a (context-dependent) linear matrix applied to $\mathbf{q}$.

Linear Attention Let us consider further the linear *self* attention module in linear transformers, where the query, key, and values are projections of the same matrix. Specifically, 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. $W_Q, W_K, W_V \in \mathbb{R}^{d' \times d}$ are the projection matrices for computing the attention queries, keys, and values, respectively; X denotes the input representations of query tokens before t. Self attention is equal: $\text{LA}(W_V X, W_K X, \mathbf{q}) = W_V X \left(W_K X\right)^T \mathbf{q} = W_V X X^T W_K^T \mathbf{q}$. We can now consider what happens to this expression when we prepend training demonstrations to the input:

$$\text{LA}(V, K, \mathbf{q}) = W_V [X'; X] \left(W_K [X'; X]\right)^T \mathbf{q}$$

X' denotes the input representations of the demonstration tokens; and $[X'; X]$ denotes the matrix concatenation. Using simple linear algebra:

$$\left(W_V X' \left(W_K X'\right)^T + W_V X \left(W_K X\right)^T\right) \mathbf{q}$$

Importantly, the expression in parentheses decomposes into two terms, one depending only on the demonstrations and the other depending only on the zero-shot input. The part that corresponds to the input is exactly the same as W before adding the demonstrations, and we denote it $W_{\text{ZSL}} = W_V X \left(W_K X\right)^T$ for zero-shot learning. The new additive term is denoted $\Delta W_{\text{ICL}} = W_V X' \left(W_K X'\right)^T$ for in-context learning.

Linear Layer Let $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be the weight matrix of a linear layer initialized at W. Let $\mathbf{x}$ be the query input representation, and let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^{d_{\text{in}}}$ the representations of training examples. A linear layer computes the function: $f(x) = W\mathbf{x}$. One step of gradient descent yields a parameter update ΔW_{GD}, after which the modified linear function is $(W + \Delta W_{\text{GD}})\mathbf{x}$ which is structurally similar to the equation for linear self attention. We already see that if, for some mysterious reason, $\Delta W_{\text{GD}} \approx \Delta W_{\text{ICL}}$, we get a justification for the claim that (linear) attention performs gradient descent. It is worth noting that the linear layer can be part of a larger model, as long as $\Delta W_{\text{GD}} = \nabla_W F(W\mathbf{x})$, we don't need to worry about what F is. Therefore, $W\mathbf{x}$ can even be part of a transformer if we can find a connection to ΔW_{ICL}. This can be hard to justify. Authors rely on empirical evidence to a certain degree (e.g., (Dai et al., 2023)) and some also have simplifying assumptions, like linear transformers (Von Oswald et al., 2023; Akyürek et al., 2023).

3 Layer Causality

We characterize a core problem with ICL-GD correspondence in the following statement:

Layer Causality In ICL, the update to the output of the l-th attention layer is dependent only on the output of previous (lower) layers. In contrast, the update to the l-th attention output induced by finetuning is determined by the gradient of the entire model's trainable parameters.

Motivated by this observation, we propose to use a layer causality-compatible finetuning method, where each layer is updated individually. Specifically, we project the output of each layer onto logits in the vocabulary space using the pretrained projection head (the unembedding head) and compute the cross-entropy loss of this prediction with respect to the one-hot embedding of the next token. This mimics regular finetuning which optimizes next-token prediction, but instead of projecting the last hidden state to the vocabulary, the current hidden state is projected – this is called *early exit* (Teerapittayanon et al., 2017). Unlike finetuning, it does not violate the causal structure of the network, as it depends only on data available in this layer. Early exit can be justified by the residual stream hypothesis (nostalgebraist, 2020), which stipulates that language models refine the next token prediction throughout the layers, and projecting internal layers into the vocabulary space gives the current prediction in this layer.

Denote the unembedding projection by $U(\cdot)$, which normally takes the final hidden state of the model and projects it onto the logits over the vocabulary. Let the detached hidden states after the ℓ-th attention layer at token i be denoted:

$$\hat{h}_i^\ell = \text{Attn}\left(W_V \text{SG}(X^\ell), W_K \text{SG}(X^\ell), \text{SG}(\mathbf{q}_i^\ell)\right)$$

where $\text{SG}(\cdot)$ stands for the "stop gradient" operation (also called `.detach()` in PyTorch) which does not affect the forward pass, but in the backward pass it does not back-propagate the gradient

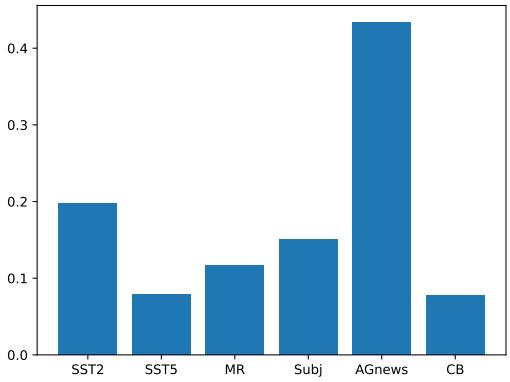

Figure 2: α averaged over all layers for each task. Computed for one seed per task.

to its input, meaning it is treated as a constant. Let the tokens of the model be represented by a list of one-hot vectors $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T$. Then, the objective function is:

$$\mathcal{L} = \sum_{i=1}^{T-1} \sum_{\ell=1}^{L} \text{CE}\left(U(\hat{h}_i^\ell), \mathbf{e}_{i+1}\right) \quad (1)$$

We assume U to be frozen as well. CE is cross-entropy loss. We optimize by taking steps with respect to the gradient $\nabla_W \mathcal{L}$, where the "stop gradient" operator makes sure each layer is updated independently. In Appendix B, we give some preliminary motivation for this method.

Relevance to Linear Transformers Linear transformers compute multivariate polynomials of the input[1] (as a composition of LSA modules, which have this property). Polynomials are not affected by layer causality since they can always be (theoretically) written in the form: $\sum_{i_1,i_2,..,i_N} \alpha_{i_1,i_2,...,i_N} x_1^{i_1} x_2^{i_2} \cdots x_N^{i_N}$ (for some exponentially large N) – where there's no concept of depth. Clearly, we can't expect the model implement it's own backward pass, but (Von Oswald et al., 2023) have shown (empirically) that a variant of GD they call GD^{++} aligns with in-context learning in the case of a linear regression task. Despite this, we do not expect this to generalize, due to the limitations of the setup and layer causality.

[1]Assuming for simplicity that input length is constant, otherwise it's a family of polynomials, one per input length.

4 Experimental Framework

Attention *Output* Similarity (SimAOU)

We adopt similarity metrics from (Dai et al., 2023) to compare the behavior of ICL and finetuning. The first metric quantifies the similarity between two updates to the attention *output* of a layer compared to the zero-shot setting. For a given query example, let $h_X^{(l)}$ represent the output representation of the last token at the l-th attention layer within setting X. The updates induced by ICL and finetuning are given by $h_{\text{ICL}}^{(l)} - h_{\text{ZSL}}^{(l)}$ and $h_{\text{FT}}^{(l)} - h_{\text{ZSL}}^{(l)}$, respectively. The attention output similarity (SimAOU) is defined as the cosine similarity between these updates, averaged across all layers. A higher SimAOU score indicates that ICL is more inclined to adjust the attention output in the same direction as finetuning. For the sake of comparison, they compare with a baseline of SimAOU with random attention output updates: $h_{\text{rand}}^{(l)} - h_{\text{ZSL}}^{(l)}$ where $h_{\text{rand}}^{(l)}$ is sampled uniformly.

It turns out that the original work (Dai et al., 2023) considered a slight variation of this, where they *normalize* $h_X^{(l)}$ for $X \in \{\text{ZSL}, \text{ICL}, \text{FT}\}$ – we call this metric SimAOU$_{\text{norm}}$. We will show this metric is misleading. In the original setting, (Dai et al., 2023) have shown that random noise gets minuscule score on this metric. However, we show this is a function of its scale. Let $\mathbf{z} = h_{\text{ZSL}}^{(l)}$ be the *unnormalized* attention output in zero-shot. Assume $\mathbf{r}, \mathbf{r}' \sim \mathcal{N}(0, \sigma I)$ are random gaussian noise vectors with variance σ^2. Now, choose σ such that $\|\mathbf{z}\| = \|\mathbf{r}\| = \|\mathbf{r}'\|$ and set $\mathbf{z}_{\text{ICL}} = \mathbf{z} + \mathbf{r}, \mathbf{z}_{\text{FT}} = \mathbf{z} + \mathbf{r}'$. The random vectors are approximately uncorrelated with each other and with $\mathbf{z}$, that is: $\mathbf{z}^T\mathbf{r} = \mathbf{r}^T\mathbf{r}' = \mathbf{z}^T\mathbf{r}' = 0$. Therefore,

$\|\mathbf{z}_{\text{ICL}}\| = \|\mathbf{z}+\mathbf{r}\| = 2\|\mathbf{z}\| = \|\mathbf{z}+\mathbf{r}'\| = \|\mathbf{z}_{\text{FT}}\|$. We get that $\text{SimAOU}_{\text{norm}}$ equals:

$$\frac{\frac{\mathbf{z}_{\text{ICL}}}{\|\mathbf{z}_{\text{ICL}}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}}{\left\|\frac{\mathbf{z}_{\text{ICL}}}{\|\mathbf{z}_{\text{ICL}}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}\right\|} \cdot \frac{\frac{\mathbf{z}_{\text{FT}}}{\|\mathbf{z}_{\text{FT}}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}}{\left\|\frac{\mathbf{z}_{\text{FT}}}{\|\mathbf{z}_{\text{FT}}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}\right\|} =$$

$$\frac{\frac{\mathbf{z}+\mathbf{r}}{2\|\mathbf{z}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}}{\left\|\frac{\mathbf{z}+\mathbf{r}}{2\|\mathbf{z}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}\right\|} \cdot \frac{\frac{\mathbf{z}+\mathbf{r}'}{2\|\mathbf{z}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}}{\left\|\frac{\mathbf{z}+\mathbf{r}'}{2\|\mathbf{z}\|} - \frac{\mathbf{z}}{\|\mathbf{z}\|}\right\|} =$$

$$\frac{\mathbf{r}-\mathbf{z}}{\|\mathbf{r}-\mathbf{z}\|} \cdot \frac{\mathbf{r}'-\mathbf{z}}{\|\mathbf{r}'-\mathbf{z}\|} = \frac{\|\mathbf{z}\|^2}{2\|\mathbf{z}\| \cdot 2\|\mathbf{z}\|} = \frac{1}{4}$$

The main property we used was the fact that after normalization $\mathbf{z}$ terms don't cancel out completely and interact with each other. This is a general problem. We compare unnormalized SimAOU with $\text{SimAOU}_{\text{norm}}$ in Table 1 and show it has substantial impact on the similarity scores.

Attention Map Similarity

SimAM is used to measure the similarity between attention maps for ICL and finetuning. For a query example, let $m_X^{(l,h)}$ represent the attention weights before softmax in the h-th head of the l-th layer for setting X. In ICL, we focus solely on query token attention weights, excluding demonstration tokens. We calculate the cosine similarity between $m_{\text{ICL}}^{(l,h)}$ and $m_{\text{FT}}^{(l,h)}$ to obtain SimAM.

One complication with SimAM is that in layer causal GD, gradients applied to lower layers have a large magnitude. On the other hand, during vanilla finetuning the gradient of early layers, which is propagated throughout the model, usually has its norm decreased. This is shown in Appendix A. Figure 4 shows the norms of each method's gradients and supports the argument empirically. We then use this realization to narrow the SimAM gap between layer causal and vanilla GD. As a preliminary solution, we suggest gradient clipping in the layer causal variant, to help mitigate gradient explosion. In the exploration provided in the Appendix, we use a manually selected clip value for **Subj**. Results are shown in Table 3.

Attention Map *Update* Similarity In light of the discussion above, we depart from (Dai et al., 2023) and suggest a modified metric, SimAM_Δ, which computes the cosine similarity between $m_{\text{ICL}}^{(l,h)} - m_{\text{ZS}}^{(l,h)}$ and $m_{\text{FT}}^{(l,h)} - m_{\text{ZS}}^{(l,h)}$, the *update* vectors. This way, the metric is less sensitive to the size of the update vector. Without it, the cosine similarity might be dominated by $m_{\text{ZS}}^{(l,h)}$ and the model drifting further from $m_{\text{ZS}}^{(l,h)}$ will be penalized even if the update direction is more similar to ICL's. The update size is controllable by adjusting the learning rate, and so is not core to the comparison between GD and layer causal GD (in our experiments we don't tune the learning rate and just use the one from GD, thus giving it an advantage). For most of the paper we will use this metric instead of the original one for the reasons mentioned above.

5 Experiments

5.1 Datasets

Following (Dai et al., 2023), we use six datasets for our experiment: **SST2** (Socher et al., 2013) **SST5** (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. Data statistics are provided in Table 4 (Appendix C).

5.2 Layer Causality

We use the same GPT-like pre-trained language models used by (Dai et al., 2023) with 1.3B implemented in fairseq.[2] We test vanilla and layer causal GD in terms of their similarity to ICL with the four variants we discussed in the previous Section. For better results, we average across 3 different seeds. The computation took 12 hours on one Tesla V100 GPU. Table 1 shows both variants of SimAOU and SimAM for both methods.

Overall, with the exception of **AGNews**, layer causal GD is significantly more aligned with ICL in terms of the modified similarity metrics, and also the normalized variant of SimAOU. However, it is important to note that the modified metrics are low for both variants.

5.3 Does Layer Causal GD Approximate GD?

A natural question that might arise is how similar GD is to the suggested layer causal method. Due to their relatively similar scores, one might conjecture that layer causal GD is a low-resource approximation for GD. We can gauge how similar the two update vectors are to each other using a variant of the attention map metric: $\text{SimAM}_\Delta^{\text{GD, LCGD}} = \text{CosSim}\left(m_{\text{LCGD}}^{(l,h)} - m_{\text{ZS}}^{(l,h)}, m_{\text{GD}}^{(l,h)} - m_{\text{ZS}}^{(l,h)}\right)$. This way we can measure how much of the score is

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

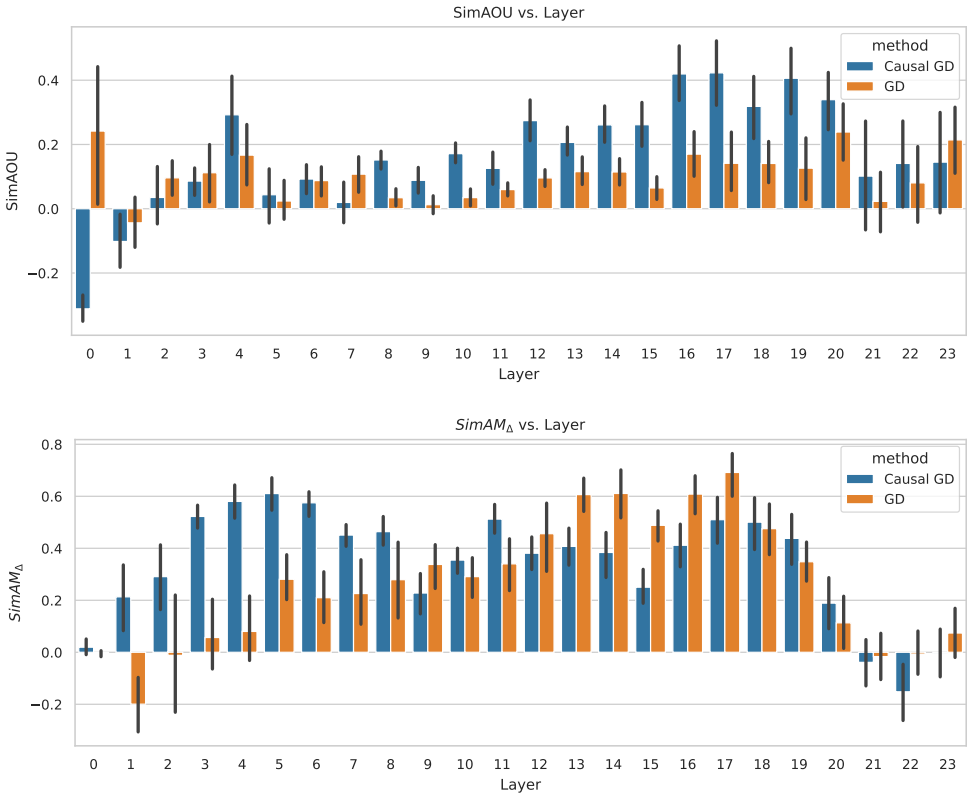

Figure 3: Similarity computed per layer aggregated across tasks and seeds. Error bar is presented. Blue bars represent layer causal GD and orange is used for vanilla GD. *Top*: SimAOU of each layer's update vector. *Bottom*: SimAM_Δ of each layer's update vector.

attributable to the similarity between the update vectors. We will denote the metric by α.

We take one seed per task and compute the average α over the layers, for each task. Counter to our expectations, Figure 2 shows that for most datasets, $\alpha \approx 0.1 - 0.2$, which is very low. This shows that the updates are *not* very correlated, and most of the score of either of the procedures *cannot be attributed to a common direction*.

5.4 Layerwise Analysis

The reported metrics until now are averaged across all layers. However, it is interesting to look at similarity patterns across layers. In Figure 3, we show the SimAOU and SimAM_Δ scores averaged across all tasks and seeds for each layer. Interesting patterns emerge in the plots. First, we notice that our variant outperforms vanilla GD in terms of SimAOU (except for layers 1, 3, and the last layer). In the second plot, we have a more complicated case. In the first half of the model, SimAM_Δ is greater for the causal variant (except for layer 9). However, for all layers 12-17, vanilla GD is substantially greater than layer causal. Beginning from layer 18, both scores decrease more or less together.

With this discrepancy between the metrics, it is worthwhile to discuss their different roles. SimAOU captures the similarity to ICL's *hidden states*. They have a direct effect on the model's prediction. Attention logits on the other hand only modulate the coefficients that determine the hidden state. The hidden state mediates their interactions with the rest of the model. They have no direct effect on the prediction, conditioned on the hidden state. On the other hand, attention maps can provide us insight into the way attention has shifted as a response to the parameter update. The higher this metric is, the better it replicates the way ICL attends to its input. While not directly affecting the output, it focuses on what "interests" ICL.

Finally, it is important to remember that our GD variant was selected intentionally due to its simplicity. Mild modifications might make it a better contender. Moreover, the setting we consider is limited to the one chosen by (Dai et al., 2023), including reusing the same hyperparameters for both methods. It is possible that tuning the hy-

perparameters for our variant would have yielded better results. All in all, we can state rather confidently that even this simple baseline *performs on par with vanilla GD* across multiple benchmarks, and in some cases outperforms it. Furthermore, it has appealing features, such as being low resource, simple, and causally plausible.

5.5 Similarity in Untrained Transformers

We have discussed variants of GD in trained models. We now turn to untrained models that are not supposed to demonstrate in-context learning. This comparison is important in order to understand to what extent the similarity can be attributed to learned parameters and what is attributable to the inductive biases of the architecture and the data itself. In Table 2, we analyze the scores of two baselines: a completely untrained model (No Training) and a model where we kept the input and output embeddings (including positional embeddings) and layer norms (Trained Embeddings). Our results paint a striking picture. We find that in terms of SimAOU the untrained baselines are *on par* and even exceed vanilla GD. It is even competitive with layer causal GD on a trained model. This means that structure of the data enables SimAOU similarity to some extent. We now can confidently say that SimAOU was low for the trained model, as an untrained model achieves the same score.

Much of the literature on ICL-GD correspondence relies on tailored weight constructions. These constructions are supposed to be learned during pretraining. Note that such constructions are *necessary* in order to "break" layer causality and encode gradient descent in the weights of a transformer.

SimAM_{Δ} has a better case, where there seems to be some gain from pretraining in the similarity score, though results are mixed. It stands to reason, since the model has learned important skills like semantic information that is encoded in both the embeddings and the model's layers. This information can be used to identify which parts of the sequence to pay attention to. It is noteworthy that just keeping the embeddings doesn't seem to improve the score, which makes sense, as the layers are also expected to play a role in extracting the information.

Due to high variance of the similarity scores, it's to hard to make a definite statement about the dominance of one variant over the other (except for LCGD SimAM_{Δ}), but it is plain that the scores are on par with each other. The evidence for a non trivial SimAOU similarity between ICL and GD variants is weak or non-existent.

6 Discussion and Limitations

In this section, we discuss future directions and limitations of our work. We discuss problems with the metrics and the datasets, as well as promising directions for future work and the main takeaways we wish to emphasize.

Benchmark: We use the same benchmark used in the original paper by (Dai et al., 2023) to make sure we do not introduce factors that benefit our method unintentionally. The set of datasets needs to be diversified. Four out of six datasets are sentiment classification datasets. One of the other tasks, CB, is very small, which contributes to instability. Similarly, we consider a specific model in all our experiments. In order to make a general claim, other models should be tested too.

Evidence Against GD-ICL: Overall, we find reasons to doubt that learning makes ICL and GD more aligned. In fact, we see evidence that an untrained model picks up approximately the same signals that a fully trained model does in terms of attention output score, and isn't far behind in terms of attention maps, hinting that these are indeed just shallow features, such as template repetition. These features seem to be identifiable even with untrained transformers due to its inductive bias.

Layer Causality: We see some evidence that layer causal GD performs better than the vanilla version which is on par with untrained baselines in terms of ICL-GD alignment. Our layer causal variant was simple by design. Multiple different design choices can be made, such as applying each layer's parameter update before moving to the next and trying different loss functions.

7 Related Work

Recent works explore the similarities between ICL and gradient descent-based optimization. (Akyürek et al., 2023) show that transformer-based in-context learners can implicitly implement standard optimization algorithms on linear models. (Von Oswald et al., 2023) demonstrate how a linear attention-only transformer model can perform a gradient descent-like procedure implicitly. (Irie et al., 2022) rewrite the dual form of a linear perceptron in terms of query-key-value attention, and use it to analyze how a trained model is affected by

		SST2	SST5	MR	Subj	CB
SimAOU	Trained	$0.05_{\pm 0.01}$	$0.04_{\pm 0.02}$	$0.17_{\pm 0.03}$	$0.06_{\pm 0.01}$	$\mathbf{0.11}_{\pm 0.01}$
	Trained Embeddings	$\mathbf{0.11}_{\pm 0.02}$	$\underline{0.06}_{\pm 0.00}$	$\mathbf{0.24}_{\pm 0.00}$	$\mathbf{0.20}_{\pm 0.00}$	$0.01_{\pm 0.00}$
	No Training	$0.09_{\pm 0.00}$	$\underline{\mathbf{0.07}}_{\pm 0.03}$	$0.18_{\pm 0.03}$	$0.06_{\pm 0.01}$	$0.04_{\pm 0.01}$
	Trained LCGD	$\mathbf{0.13}_{\pm 0.01}$	$\mathbf{0.11}_{\pm 0.01}$	$0.21_{\pm 0.03}$	$0.18_{\pm 0.00}$	$\mathbf{0.24}_{\pm 0.01}$
SimAM_{Δ}	Trained	$\mathbf{0.15}_{\pm 0.02}$	$\mathbf{0.31}_{\pm 0.02}$	$0.14_{\pm 0.05}$	$\mathbf{0.25}_{\pm 0.07}$	$\underline{0.25}_{\pm 0.01}$
	Trained Embeddings	$0.09_{\pm 0.02}$	$0.03_{\pm 0.00}$	$\mathbf{0.18}_{\pm 0.02}$	$0.16_{\pm 0.02}$	$0.05_{\pm 0.10}$
	No Training	$0.11_{\pm 0.04}$	$0.05_{\pm 0.03}$	$0.16_{\pm 0.03}$	$0.17_{\pm 0.03}$	$\underline{\mathbf{0.26}}_{\pm 0.05}$
	Trained LCGD	$\mathbf{0.30}_{\pm 0.02}$	$\mathbf{0.33}_{\pm 0.01}$	$\mathbf{0.26}_{\pm 0.00}$	$\mathbf{0.32}_{\pm 0.01}$	$\mathbf{0.38}_{\pm 0.01}$

Table 2: SimAOU and SimAM comparison of vanilla GD for trained and untrained transformers. When the difference between the highest and second-highest score in a column is ≤ 0.01, we underline both scores. We also add a row for LCGD for comparison, which we write in boldface if it's the highest score, however, we still write the highest score *excluding* LCGD in boldface.

its training samples.

Unlike these works, (Dai et al., 2023), which we are heavily influenced by, study large GPT transformers on structured language classification tasks. We study how different aspects of ICL can affect the results of the comparison made in (Dai et al., 2023). While (Dai et al., 2023) compare standard GD-based finetuning, we test a layer causal finetuning procedure. Recently, new works have emerged (Todd et al., 2023; Hendel et al., 2023) suggesting a different approach to interpreting ICL as an algorithm that compresses training demonstrations into a function/task vector that steers the model to perform the task. Other explanations of ICL from other perspectives include induction heads (Olsson et al., 2022) and Bayesian inference (Xie et al., 2022).

Most similar to our paper is the concurrent work of (Shen et al., 2023), which points to another discrepancy between *full batch* GD and ICL. They show that vanilla full batch GD and ICL cannot be reconciled due to ICL's sensitivity to the order of the demonstrations, which full batch GD is invariant to. However, this discrepancy can be mitigated easily by applying GD sequentially, as was done in the work of (Dai et al., 2023) that we compare to.

Layer causal GD is similar to (Bengio et al., 2006) where a similar idea was proposed to accelerate training by finding a good starting point using a greedy layer-wise approach.

8 Conclusions

Inspired by recent work, we attempt to explore the relationship between in-context learning and gradient descent-based finetuning in practical settings. We address a fundamental difference in information flow between the methods and suggest a variant that respects layer causality. Contrary to common belief, we find no evidence of that ICL and GD are similar in the latent space. We show that untrained models perform on par with the pretrained model in terms of similarity scores. We hope that parts of the theory can still be salvaged. For example, the results of (Von Oswald et al., 2023) seem to be robust in the restricted case they consider. This can be the object of future exploration. There might be other metrics that represent a subtler similarity than those considered in this paper.

References

Mark A. Aizerman, E. M. Braverman, and Lev I. Rozonoer. 2019. Theoretical foundation of potential functions method in pattern recognition.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

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.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse.

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

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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9639–9659. PMLR.

nostalgebraist. 2020. interpreting gpt: the logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, T. J. Henighan, Benjamin Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, John Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom B. Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Christopher Olah. 2022. In-context learning and induction heads. *ArXiv*, abs/2209.11895.

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 on Association for Computational Linguistics*, ACL '04, page 271–es, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 115–124, USA. Association for Computational Linguistics.

Lingfeng Shen, Aayush Mishra, and Daniel Khashabi. 2023. Do pretrained transformers really learn in-context by gradient descent?

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.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2017. Branchynet: Fast inference via early exiting from deep neural networks.

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

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.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference.

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.

A Additional Material

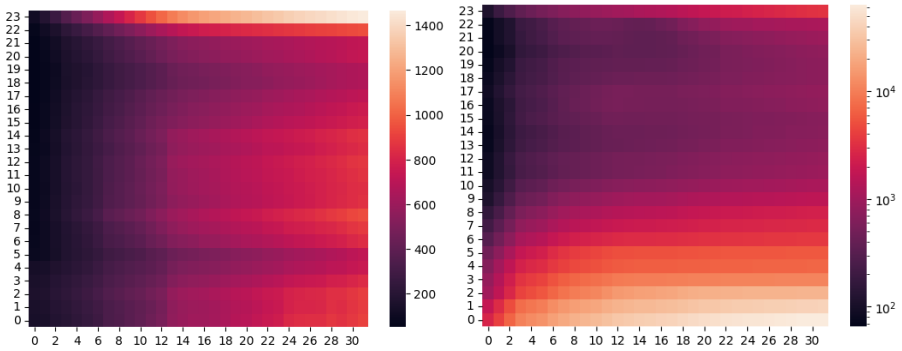

Figure 4: Heatmap of ℓ_2 norms of the gradients computed during finetuning on the **Subj** task. Note the different scales of magnitude. **Horizontal Axis**: training demonstration index. **Vertical Axis**: layer index in ascending order (from input to network output). **Left**: Vanilla GD. **Right**: Layer Causal GD (norm magnitude in logarithmic scale).

	Subj
SimAUO (GD)	0.1932
SimAUO (LCGD Clipped)	**0.3480**
SimAM (ZSL)	0.3786
SimAM (GD)	**0.4870**
SimAM (LCGD Clipped)	0.4227

Table 3: Comparison of layer causal GD with gradient clipping. These results show that even arbitrary clipping may resolve the drop in SimAM shown in Table 1.

B Heuristic "Derivation" of Objective

In this section, we give some heuristic motivation for Eq. (1). For simplicity, we will focus on the case of one demonstration and a test example. First, a simple observation. Let us consider the output of the attention layer of a token in the test example. We will assume a single attention head. We want to express it in terms of two expressions, $\mathbf{o}_i \in \mathbb{R}^d$, the (zero-shot) output of the i-th attention layer at the test token *without* the demonstration, and $\tilde{\mathbf{o}}_i \in \mathbb{R}^d$ the cross-attention output between the token and the demonstration at layer i:

$$\text{softmax}(\left[\mathbf{q}_i^\top K_{demo}; \mathbf{q}_i^\top K_{test}\right])\,[V_{demo}; V_{test}]\,W_O =$$
$$\alpha_i\ \text{softmax}(\mathbf{q}_i^\top K_{demo})V_{demo}W_O + (1-\alpha_i)\ \text{softmax}(\mathbf{q}_i^\top K_{test})V_{test}W_O =$$
$$\alpha_i\ \tilde{\mathbf{o}}_i + (1-\alpha_i)\mathbf{o}_i = \mathbf{o}_i - \alpha_i(\mathbf{o}_i - \tilde{\mathbf{o}}_i) = \mathbf{o}_i - \frac{\alpha_i}{2}\nabla_{\mathbf{o}_i}||\mathbf{o}_i - \tilde{\mathbf{o}}_i||^2 =$$
$$\mathbf{o}_i - \alpha_i \nabla_{\mathbf{o}_i}\ell\left(\mathbf{o}_i, \tilde{\mathbf{o}}_i\right)$$

Where $\ell(\mathbf{o}_i, \tilde{\mathbf{o}}_i) = \frac{1}{2}||\mathbf{o}_i - \tilde{\mathbf{o}}_i||^2$ is the squared error loss. The second line comes from the fact that softmax is almost an elementwise operation, except for a normalization factor, and it is easy to see that the softmax of a concatenation of vectors is the concatenation of the softmax vectors where each is scaled by a normalization factor. Since the vector still needs to sum up to one, the coefficients need to sum up to one as well. We denote these factors by α_i and $1-\alpha_i$. The next step follows from definition, and the rest is computation. This statement is tautological and it holds in every attention layer. It is noteworthy that we made the attention update look like a gradient step. Also, notice that it is very strongly contingent on stopped gradients. However, $\mathbf{o}_i$ is not a *parameter*, but rather an intermediate computation step. To have an optimization process, we need a parameter update. We approximate this process by replacing it with

gradient descent on the current attention layer. We will motivate this presently. It is important for us to explain two parts: Why should we expect that gradient descent on the attention layer can approximate this shallow gradient step? And why only this attention layer and not other layers?

We draw intuition from the chain rule and backpropagation. A gradient step is used to shrink the error on the provided training example's loss. Interestingly, backpropagation barely makes a distinction between trainable parameters and intermediate steps. If it is a trainable parameter, it is a leaf of the gradient computation, and no further steps are required. Otherwise, the gradient continues to propagate upstream. This is because we cannot directly optimize intermediate steps like $\mathbf{o}_i$, but we can optimize parameters upstream to make this error smaller for $\mathbf{o}_i$. Gradient descent gives the best local direction to make the downstream error smaller. Unfortunately, $\mathbf{o}_i$ is not a parameter. Therefore, to minimize $(\mathbf{o}_i - \tilde{\mathbf{o}}_i)^2$ using gradient descent when $\mathbf{o}_i$ is a parameter, the next best thing is to continue the propagation to the trainable parameters.

On the other hand, as discussed throughout the paper, we cannot let the gradient propagate to previous layers. The existence of the demonstration examples affects the computation only through the attention map, which is the only part that mixes information between tokens.

So, putting all parts together, we gave some substance to the claim that the model is expected to make a layer causal gradient step with a per-parameter learning rate α_i, which also changes in every step. This is not different from some other optimization algorithms and schedules that change the learning rate across these two axes as well. However, note that the loss function is different from Eq. (1). Also, when there is more than one demonstration, the argument becomes even more delicate.

C Data Statistics

	# Train	# Validation	Avg. # of Tokens
SST2	67,349	1,821	55.43
SST5	8,544	2,210	102.95
MR	8,530	1,066	113.39
Subj	8,000	2,000	129.23
AGNews	120,000	7,600	237.72
CB	250	250	295.80

Table 4: Data statistics of all the datasets in the benchmark