

ICLM: An Interpretable Model System for the Transformer

Anonymous submission

Abstract

We propose a new generative model with a similar structure to the Transformer but designed for interpretability. Our model decomposes next-token prediction into two distinct functional components: (1) a *feature extractor* that encodes the contextual token sequence into real-valued covariates, and (2) a *kernel-based function* that uses these covariates to generate a distribution over tokens. This decomposition enables a transparent view of inference as a single or multi-step functional gradient-descent update. We demonstrate that our model achieves competitive performance with the Transformer on standard language modeling benchmarks using medium sized models of equal parameter count, and show that it parallels many of the Transformers mechanistic behaviors. We leverage these similarities to argue that our new model can serve as an effective *model system* for analyzing the Transformer. Our framework offers a fresh theoretical and practical perspective on Transformer-based language modeling, with implications for interpretability, model understanding, and safety.

1 Introduction

Language models based on the Transformer have become foundational to modern natural language processing (NLP) [Vaswani et al. 2017, Brown et al. 2020, DeepSeek 2025]. Despite their empirical success, Transformers are notoriously complex, and achieving a clear mechanistic understanding of their behavior remains an open challenge. This opacity poses significant barriers to interpretability, predictability, and trust.

There have been two recent lines of work aimed at demystifying the Transformer’s internal mechanisms: (i) **Mechanistic Studies** that directly analyze Transformer components in the context of language modeling, often done by “freezing” or removing specific components to isolate the contributions of those remaining active [Dong et al. 2025, Olsson et al. 2022a, Chen et al. 2025]; and (ii) **Theoretical Investigations** that offer a principled framework for interpreting Transformer behavior, but consider simpler, *non-language* data or non-standard variants of the Transformer [von Oswald et al. 2023, Cheng, Chen, and Sra 2024, Wang et al. 2025].

We seek to bring these independent research directions together for the first time by introducing a model with a strong

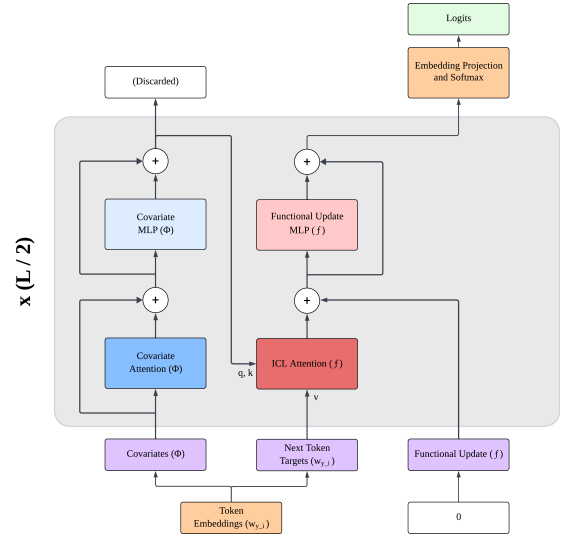


Figure 1: Summary of the ICLM model. Covariates and next-token targets are initialized with the same embedding vectors, but their positions are offset by one. We use q , k , and v to represent the source of the queries, keys, and values in the ICL attention blocks. For consistency with the Transformer, we define the number of layers, L as the total number of attention/MLP pairs, and thus we have $L/2$ updates for both ϕ and $f(\phi)$.

theoretical grounding that we can use to isolate mechanistic elements of importance for language modeling.

In the study of human biology, it is common to introduce interpretable *model systems* [Kim, Koo, and Knoblich 2020] – simpler organisms that, while less complex than the human systems they approximate, offer critical mechanistic insights through their interpretability and transparency. An effective model system is one that is easy to analyze while remaining as representative of the original system as possible. Our model is well-understood and built on a substantial amount of prior research, making it a good subject for mechanistic studies. Furthermore, our experiments show that our model has strong language modeling capabilities and that our chosen decomposition aligns well with recent Transformer mechanistic studies. Until now, this combination of performance and interpretability has been difficult to achieve. Our new model is a first attempt at developing a strong model system for the Transformer, and we hope that it will be a useful tool for future mechanistic studies.

An overview of our model is shown in Figure 1. We call it the “In-Context Learning Model,” or ICLM, due to the in-context gradient-descent meta-learning mechanisms that fuel its predictive capabilities. It consists of a token embedding layer, multiple attention/MLP pairs, and a final projection followed by softmax. This formulation is similar to the original Transformer [Vaswani et al. 2017], but incorporates a stronger inductive bias.

Leveraging ideas from the theoretically-grounded ICL literature [Cheng, Chen, and Sra 2024, Wang et al. 2025], we decompose the attention and MLP layers into two distinct functional components: a **feature extraction function** ϕ that maps preceding tokens (context) into a vector of covariates, and a **predictive function** f , defined as a kernel-based function of these covariates. This yields a model that is consistent with prior ICL work, but extended to operate effectively as a language model.

Contributions

- We propose a new generative model, the ICLM, that decomposes language modeling into two distinct parts: feature extraction and kernel-based prediction.
- We conduct extensive experiments to demonstrate that the ICLM model is a faithful “model system” for the Transformer by benchmarking its language modeling capabilities across various model sizes.
- We provide several potential applications of the ICLM model during mechanistic studies of the Transformer
- We provide theoretical insights into how this model relates to previous ICL work with Transformers that may fuel future language-based ICL-inspired models.

Related Work

Transformer Interpretability. Many researchers have attempted to analyze the internal representations and mechanisms of Transformers. Vig et al. [2020], Clark et al. [2019], Xiao et al. [2024] study the role of attention heads, while Elhage et al. [2021] conduct mechanistic analyses of Transformer circuits. There has also been work on induction heads [Olsson et al. 2022a] for language ICL, and on investigation of the MLP elements for memory [Geva et al. 2021, Dong et al. 2025, Chen et al. 2025].

In-Context Learning and Function Approximation. Olsson et al. [2022b] and Garg et al. [2022] suggest that Transformers perform a form of gradient descent during in-context learning. Concurrently, Akyürek, Andreas, and Lin [2022] show that Transformers can approximate gradient-based learning algorithms. Most prior work of this type [von Oswald et al. 2023, Cheng, Chen, and Sra 2024, Wang et al. 2025] has considered *functional* data, with data pairs (x_i, y_i) , where x_i is a function input and y_i is the output.

The above two streams of related research have been undertaken largely independently. For the first time, we seek to bring them together, leveraging the strengths of each. After presenting the details of our framework, in Section 6 we make explicit connections to this related prior work.

2 Two Underlying Functions for Language

Consider a vocabulary $\mathbb{V} = \{1, \dots, V\}$, with $y_i \in \mathbb{V}$ representing the i th token in a sequence of text, y_1, \dots, y_N . We also consider a distinct token y_0 , which is at the start of each text sequence. In a generative language model, interest is typically in predicting the next token y_{N+1} conditioned on the context of the preceding N observed tokens. Following standard autoregressive modeling [Vaswani et al. 2017], we assume

$$p(y_1, \dots, y_N) = \prod_{i=1}^N p(y_i | y_0, \dots, y_{i-1}). \quad (1)$$

We assume each token i has a corresponding real-valued covariate $\phi_i \in \mathbb{R}^d$ that summarizes the preceding context (y_0, \dots, y_{i-1}) . Formally,

$$\phi_i = g(y_0, \dots, y_{i-1}), \quad (2)$$

where $g : \mathbb{V}^i \rightarrow \mathbb{R}^d$ is a (learned) feature extraction function. This step transforms discrete token sequences into continuous representations. Crucially, ϕ_i serves as a context-dependent covariate: it encapsulates the relevant predictive information from the token prefix. In this sense, ϕ_i acts as a sufficient statistic for predicting y_i . This abstraction enables a clean separation between contextual encoding and prediction, facilitating interpretability and modular design.

We then model the conditional distribution of y_i given the context (y_0, \dots, y_{i-1}) using a softmax over a context-dependent function f applied to ϕ_i :

$$q(y_i | \phi_i) := \frac{\exp(w_{y_i}^\top f(\phi_i))}{\sum_{v=1}^V \exp(w_v^\top f(\phi_i))}, \quad (3)$$

where $q(y_i | \phi_i) = p(y_i | y_0, \dots, y_{i-1})$, $w_v \in \mathbb{R}^d$ is the learned embedding vector for token v (independent of context), and $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a function learned in-context from observed data.

The entire prediction mechanism thus factorizes into two modular components:

1. **Feature Extraction:** Compute $\phi_i = g(y_0, \dots, y_{i-1})$ for each $i = 1, \dots, N + 1$.
2. **In-Context Function Learning:** Given (ϕ_i, y_i) for $i = 1, \dots, N$, learn the predictive function f to maximize the likelihood under (3), and use that inferred function on $f(\phi_{N+1})$ to predict the next token, y_{N+1} .

After representing the contextual data in the form $\{(\phi_i, y_i)\}_{i=1, N}$ with a query ϕ_{N+1} , this setup aligns with prior Transformer ICL work with functional data [von Oswald et al. 2023, Cheng, Chen, and Sra 2024, Wang et al. 2025], which we elucidate on next.

3 Predictive Function via Functional GD

We now describe how the predictive function $f(\phi)$ is learned in context. Function $f \in \mathcal{F}$ is viewed as the minimizer of an empirical loss, optimized in a restricted function space \mathcal{F} via gradient descent. We show that this update process closely resembles multi-head self-attention in a Transformer.

Using (1) and the definition of $q(y_i | \phi_i)$ in (3)

$$p(y_1, \dots, y_N) = \prod_{i=1}^N q(y_i | \phi_i). \quad (4)$$

For this section, we assume that $\phi_i = g(y_0, \dots, y_{i-1})$ is known, and in Section 4 we discuss how g can be learned by a Transformer. Note that we model the observed tokens (y_1, \dots, y_N) , while the start token y_0 is used as a reference within the feature representation $g(y_0, \dots, y_{i-1})$.

In-context Cross Entropy Minimization

Given a sequence of tokens (y_1, \dots, y_N) and their associated covariates (ϕ_1, \dots, ϕ_N) , we define the empirical cross-entropy loss as a functional of f :

$$\mathcal{L}(f) = - \sum_{i=1}^N \log \left[\frac{\exp(w_{y_i}^\top f(\phi_i))}{\sum_{v=1}^V \exp(w_v^\top f(\phi_i))} \right] \quad (5)$$

The in-context optimal function is the minimizer of this loss: $\hat{f} = \arg \min_{f \in \mathcal{F}} \mathcal{L}(f)$. To impose structure and tractability, we restrict the function class \mathcal{F} to be a reproducing kernel Hilbert space (RKHS) [Schölkopf and Smola 2002]. Specifically, each $f \in \mathcal{F}$ is expressed as a weighted sum over feature maps:

$$f(\phi) = \sum_{h=1}^H A_h \psi_h(\phi) \quad (6)$$

where $\psi_h(\phi) : \mathbb{R}^d \rightarrow \mathbb{R}^{d_\psi}$ are feature maps into the RKHS, and d_ψ is the dimension of the Hilbert space (in principle d_ψ could be infinity), and $A_h \in \mathbb{R}^{d' \times d_\psi}$ reflect *context-dependent* parameters. Such an assumption is consistent with recent work [Cheng, Chen, and Sra 2024, Wang et al. 2025]. As discussed next, this leads to the representation of $f(\phi)$ as a linear combination of H kernels, where kernel h is $\kappa_h(\phi_i, \phi_j) = \psi_h^\top(\phi_i) \psi_h(\phi_j)$. This will be demonstrated to connect to H attention heads in a Transformer.

RKHS Gradient Descent for learning f

Similar to the analysis in [Wang et al. 2025], we apply *functional* gradient descent (GD) on the loss in (5) to solve for \hat{f} . As derived in Appendix A, this yields the following update rule:

$$f_{i,k+1} = f_{i,k} + \sum_{h=1}^H P_h^{(k)} \sum_{j=1}^{i-1} [w_{y_j} - \mathbb{E}(w|f_{j,k})] \kappa_h(\phi_i, \phi_j) \quad (7)$$

where for kernel (head) h at gradient step $k \geq 0$, $P_h^{(k)} \in \mathbb{R}^{d' \times d'}$ is a generalized learning rate (see Appendix A), and $f_{i,k} = f_k(\phi_i)$, where $f_k(\phi_i)$ is the latent function evaluated at ϕ_i for GD step $k \geq 0$. The kernel $\kappa_h(\phi_i, \phi_j)$ quantifies context similarity, between $(y_0, y_1, \dots, y_{i-1})$ and $(y_0, y_1, \dots, y_{j-1})$, in terms of corresponding covariates ϕ_i and ϕ_j , respectively.

The expression $\mathbb{E}(w|f_{i,k})$ in (7) is the expected token embedding vector, using the softmax probability over tokens associated with $f_{i,k}$ [Wang et al. 2025]:

$$\mathbb{E}(w|f_{i,k}) = W_e \cdot \text{softmax}(W_e^\top f_{i,k}) \quad (8)$$

where $W_e \in \mathbb{R}^{d' \times V}$ is a matrix with columns corresponding to the embedding vectors of the V distinct types of tokens. In (8), $\text{softmax}(W_e^\top f_{i,k})$ represents the probability of each token given $f_{i,k}$, and multiplication of that PMF with W_e yields the expectation over embedding vectors.

Notice that for update of vector $f_{i,k+1}$, we only attend to vectors $j = 1, \dots, i-1$. This is because the contextual features ϕ_i , as discussed above, depend on tokens y_0, \dots, y_{i-1} , and we hence preserve that causality when updating the predictive function $f(\phi)$. This characteristic of our model is a result of functional GD and the nature of the associated covariates, but interestingly it aligns with the causal form of attention widely used in language models [Brown et al. 2020].

Standard RKHS theory uses symmetric kernels κ_h in (7). In Appendix B we demonstrate how this may be extended to the softmax-based attention widely used in Transformers [Vaswani et al. 2017]. The relationship between RKHS and softmax attention was also discussed in [Wang et al. 2025].

4 Model for Feature-Extraction Function

One could consider many types of neural-network-based models for $\phi_i = g(y_0, \dots, y_{i-1})$, but it is natural to consider a Transformer-based model. To be specific, for $g(y_0, \dots, y_{i-1})$ we consider a standard Transformer:

$$\begin{aligned} \tilde{\phi}_{j,\ell+1} &= \phi_{j,\ell} \\ &+ \sum_{h=1}^H \bar{P}_h^{(\ell)} \sum_{i=1}^j \bar{W}_{V,h}^{(\ell)} \phi_{i,\ell} \kappa(\bar{W}_{K,h}^{(\ell)} \phi_{i,\ell}, \bar{W}_{Q,h}^{(\ell)} \phi_{j,\ell}) \\ \phi_{j,\ell+1} &= \tilde{\phi}_{j,\ell+1} + \text{MLP}(\tilde{\phi}_{j,\ell+1}) \end{aligned} \quad (9)$$

where $\kappa(\cdot, \cdot)$ represents softmax attention [Vaswani et al. 2017], and at the input layer $\phi_{j,0} = w_{y_{j-1}} + p_{j-1}$, where p_i is a positional embedding vector. We write $\phi_{j,0}$ in terms of positional embeddings, to emphasize that token location is accounted for when constituting covariates; in our implementation we have actually employed rotary attention [Su et al. 2023]. Note that in this construction, at all layers ℓ , the covariates $\phi_{j,\ell}$ only depend on the tokens $i = 1, \dots, j-1$.

In (9), we use “bar” notation (e.g., $\bar{P}_h^{(\ell)}$) on Transformer parameters associated with the update of covariates $\phi_{j,\ell}$, to distinguish such from similar (but distinct) parameters (e.g., $P_h^{(\ell)}$) in the update equations for $f_{j,\ell}$ in (7).

While g is implemented in (9) via a standard Transformer, its **role is restricted** to generating context-dependent features ϕ , which are **used exclusively for context-similarity computations** in f . This clean separation highlights ϕ ’s role in computing context similarity, and f ’s role in generating predictions – making each component of the model interpretable in isolation.

5 A Single GD Step Across Multiple Layers

Overall model Using (9) a *sequence* of features are constituted. To map this to a *single* set of covariates, we concatenate the features from each layer to be the total set of covariates:

$$\phi_i = (\phi_{i,0}, \dots, \phi_{i,L-1}) \quad (10)$$

for L layers in a hierarchical representation of the cumulative feature vector. We will use these features within a single step of functional GD, utilizing (7) for $k = 0$.

In much prior Transformer-based ICL work [Cheng, Chen, and Sra 2024, Wang et al. 2025] the latent function in (7) is *initialized* as $f_0(\phi) = 0_{d'}$ for all ϕ , where $0_{d'}$ is an all-zeros d' -dimensional vector. Under this initialization, the expectation $\mathbb{E}(w|f_{i,0}) = (\sum_{v=1}^V w_v)/V$. Assuming this average is zero (a condition that can be encouraged via regularization or mean-centering the embedding matrix), the expectation term in (7) vanishes at $k = 0$, i.e., $\mathbb{E}(w|f_{i,0}) = 0_{d'}$.

Under these conditions, using (7) with $\mathbb{E}(w|f_{i,0}) = 0_{d'}$, one step of functional GD with the covariates in (10) is

$$f_{i,\ell+1} = f_{i,\ell} + \sum_{h=1}^H P_h^{(\ell)} \sum_{j=1}^{i-1} w_{y_j} \cdot \kappa_h(\phi_{i,\ell}, \phi_{j,\ell}) \quad (11)$$

for $\ell = 0, \dots, L-1$, with $f_{i,0} = 0_{d'}$. The index ℓ denotes the layers in the covariate Transformer (9), where $f_{i,\ell}$ for all ℓ are used to build up the function in (7) for $k = 0$.

The cumulative model embodied by (9) and (11) may now be viewed as a sequence of alternating attention blocks: In (9) the covariates are updated (independent of f) for each layer ℓ , and given the updated covariates the predictive function is updated as in (11); recall the summary in Figure 1.

Concerning interpretation, recall that f_i is employed in (3) to provide the probability of token y_i . Layer- ℓ covariates $\phi_{i,\ell}$ constitute a representation of the context (y_0, \dots, y_{i-1}) . The kernel $\kappa_h(\phi_{i,\ell}, \phi_{j,\ell})$ quantifies the similarity of contexts (y_0, \dots, y_{i-1}) and (y_0, \dots, y_{j-1}) , based on layer- ℓ features. In (11) these kernel-constituted weights are multiplied by embedding vector w_{y_j} , representative of the next token associated with context (y_1, \dots, y_{j-1}) . Therefore, f_i uses the contextual data to align itself with the embedding vector(s) for which it has the strongest match, as reflected by the kernel. The H attention heads measure the covariates from (in general) different perspectives.

6 Mechanistic Studies & Introducing MLP

Relationship to induction heads The update rule in (11) has a form that closely mirrors *induction heads* observed in attention-only Transformers [Olsson et al. 2022a]. Simply stated, induction heads are specialized attention patterns that serve the following function: Given a sequence of tokens (y_1, \dots, y_N) , induction heads find the most recent occurrence of y_N in the context, and for that match, use the observed next token to predict y_{N+1} .

Our functional GD formulation (11) formalizes this behavior through a soft, similarity-driven mechanism. Specifically, the predictive token f_{N+1} is constructed as a weighted sum over past token embeddings w_{y_j} , where the weights are given by $\kappa_h(\phi_{N+1}, \phi_j)$ for $j < N+1$. These kernel values measure similarity between covariates (ϕ_{N+1}, ϕ_j) , which in turn quantify the similarity of contexts (y_0, \dots, y_{j-1}) and (y_0, \dots, y_N) .

Introduction of the MLP for memory Recent work has shown that the MLP components of a Transformer play a key

role in modeling memory: information recalled from training data that is not directly present in the input prompt [Dong et al. 2025, Geva et al. 2022, 2021]. For example, [Dong et al. 2025] demonstrates that MLP layers are critical for solving mathematical problems that require retrieving factual knowledge.

Our functional GD formulation in (11) does not require MLP layers, and is sufficient for tasks that are *purely in-context*; this is consistent with [Olsson et al. 2022a], which (at least for small) attention-only models “present strong, causal evidence” for such phenomena. However, recognizing the emerging understanding [Geva et al. 2022, 2021, Dong et al. 2025] of the importance of the MLP to move beyond ICL, we augment (11) with an additional MLP layer:

$$\begin{aligned} \tilde{f}_{i,\ell+1} &= f_{i,\ell} + \sum_{h=1}^H P_h^{(\ell)} \sum_{j=1}^{i-1} w_{y_j} \cdot \kappa_h(\phi_{i,\ell}, \phi_{j,\ell}) \\ f_{i,\ell+1} &= \tilde{f}_{i,\ell+1} + \text{MLP}(\tilde{f}_{i,\ell+1}) \end{aligned} \quad (12)$$

In Section 7, we show empirically that the introduction of the MLP (and associated skip connection) via (12) is particularly important for performing well on mathematical tasks, corroborating [Dong et al. 2025]. We also emphasize that the role of the MLP is restricted to our final f predictions, preventing it from directly impacting future f updates.

Summary and interpretability Our ICLM architecture consists of two parallel components: (9) and (12), as illustrated in Figure 1. These reflect a *principled decomposition* of the overall function used for language generation.

The first component, defined in (9), is responsible for computing the covariates ϕ_i based on preceding context. These covariates are not used directly to predict the next token. Instead, they only serve to compute context similarity via the kernel $\kappa_h(\phi_i, \phi_j)$, which quantifies how similar two contexts $(y_0 \dots y_{i-1})$ and $(y_0 \dots y_{j-1})$ are.

The second component updates the predictive function f using the rule in (12). This update consists of a weighted sum of embedding vectors w_{y_j} , where the weights are given by the aforementioned context similarities $\kappa_h(\phi_i, \phi_j)$. The form of (12) is grounded in the functional gradient descent update (11). We again emphasize that the ϕ_j ’s only contribute to the context-similarity weights, and do not directly generate predictions.

Our model also draws on recent mechanistic insights: (1) the use of attention for context retrieval, via the induction head mechanism [Olsson et al. 2022a], and (2) the role of MLP in encoding memory [Dong et al. 2025]. In section 7, we explore the effectiveness of the model with and without the MLP unit in (12), showing (consistent with recent work [Geva et al. 2022, 2021, Dong et al. 2025]), that MLPs are especially critical for tasks involving memory, like mathematical operations.

7 Experiments

We seek to demonstrate that ICLM serves as a faithful proxy to the Transformer by examining its relative predictive performance and generative capabilities. We then present *initial* perspectives on areas where the ICLM’s structure could

assist in future mechanistic studies, namely in analyzing *attention sinks* [Xiao et al. 2024] and *induction heads* [Olsson et al. 2022a]. Finally, we provide a basic analysis of the role of the MLP within the ICLM model.

For each comparison of ICLM and the Transformer, the models were trained in the same manner, with the same number of parameters, to provide a direct comparison between the models. When the Transformer has L attention-block layers, ICLM has $L/2$ pairs of blocks of updates of ϕ and f , as depicted in Figure 1. All internal dimensions were kept the same with the sole exception of the removal of W_v heads in the ICLM models and the corresponding adjustment of the projection matrices. Our Transformer model largely follows the architecture of GPT-2 [Radford et al. 2019], but incorporates rotary embedding [Su et al. 2021] and weight tying at the output [Press and Wolf 2017]. All training and evaluation code will be released upon publication of this paper. Further details about our training process are found in the Supplemental Material (SM), along with the code.

Model Parity

When dealing with a model system, it is expected that there will be a tradeoff between interpretability and performance. Nevertheless, it is important that ICLM recover as much of the Transformer’s generative performance as possible.

Small Models Table 1 shows results for $L = 8$ and $L = 12$ layer models on relatively small corpora. We provide these results for three reasons. First, if we are to believe that the ICLM model is mechanistically similar to the Transformer, we would expect to see that it is capable of similar performance at all scales. A larger model may achieve high performance by very different means through the flexibility provided by more parameters, but a smaller model does not have this luxury. Second, we wish to see how well the model achieves parity across different types of textual data. Here we compare relatively simple narrative data with diverse and complex naturalistic data, and in subsequent sections we show results on a simple mathematical dataset.

We first consider $L = 8$ layer models trained on the TinyStories dataset [Eldan et al. 2023], synthetically generated children’s stories. We have chosen to use the updated TinyStories-V2 version [Nabeshima 2024] as the quality of the text is higher. Following [Eldan et al. 2023], we also tasked an LLM (specifically GPT-4o [OpenAI 2024]) to evaluate and score model generations on several specific rubric items. Further details about this process are provided in the SM and follows a similar analysis to [Eldan et al. 2023].

We also consider $L = 12$ layer models trained on the GoodWiki dataset [Choi 2023], a collection of curated and cleaned Wikipedia pages. We chose to use this dataset as an alternative to the commonly used Wikitext-103 dataset [Merity et al. 2016] as it is notably larger and better formatted, though the content is largely the same. Each model was then evaluated on a held-out test splits.

As summarized in Table 1, ICLM showed *very similar* performance on the GoodWiki and TinyStories data. Further, the GPT4o evaluations for results with TinyStories show that ICLM and the Transformer achieve equal average scores, but

also that ICLM scored *similarly or better* on each of the subsection scores.

Medium Models In Table 2 we show results for $L = 24$ layer ICLM and Transformer models, matching the size GPT-2 Medium [OpenAI Community]. We include W_v heads in the f update layers to keep parameter counts exactly equal, but these act directly on the in-context embeddings, as per our theory. The models were trained on a 6B token subset of SlimPajama [DKYoon], a filtered and de-duplicated web-data pretraining set chosen for its high quality and diversity of text. We calculate perplexity on a held-out test-split and evaluate the models on various language and reasoning benchmarks. We also provide benchmark results for a pre-trained GPT-2 Medium model [OpenAI Community] as reference, though differences in the training pipeline and datasets make it a less exact comparison than our own Transformer model.

As depicted in Table 2, results show a strongly competitive performance from ICLM with perplexity and accuracy scores, well within the margin of error.

Training Considerations and Scalability While the above results demonstrate close agreement between ICLM and the Transformer, note that the ICLM implementation considered here set $d = d'$, recalling $\phi \in \mathbb{R}^d$ and $f \in \mathbb{R}^{d'}$. This was done for simplicity, but one can consider other partitions of d and d' while keeping the parameter count unchanged. By contrast, the Transformer may in principle allocate its latent space in whatever configuration is optimal. Future work could explore adjusting the allocation of parameters between the d and d' to optimize performance while keeping the total number of parameters the same.

Additionally, due to limitations in our available computational resources, the majority of our experiments were conducted using small and medium sized language models. We note that the total size, memory footprint, and training time of our ICLM models is very similar to that of their Transformer counterparts, and our experience indicates that they will scale beyond the sizes we have shown.

Potential Mechanistic Applications

The decomposition of our model into two distinct updates (for ϕ and f) allows analysis of each of those updates independently and with a greater sense of understanding. Here we demonstrate two interesting consequences of this division, and how they might be used for future mechanistic research.

Attention Sinks Figure 2 shows a heatmap of the attention layers within our medium-size ICLM and Transformer models for a select prompt, on the data considered in Table 2. Within the Transformer’s attention, the manifestation of “attention sinks” has been studied [Xiao et al. 2024, Burtsev et al. 2020, Darcet et al. 2023], where high attention scores are realized for particular tokens in order to provide normalization within the softmax [Xiao et al. 2024] or otherwise help manifest global context [Burtsev et al. 2020, Darcet et al. 2023]. These sinks are often manifested in the early tokens, so they may consistently attend with the rest of the context [Xiao et al. 2024]. The ICLM shows similar

Model	GoodWiki (PPL)	TS-V2 (PPL)	TS-V2 (LLM Avg)	TS-V2 (Grammar)	TS-V2 (Consistency)	TS-V2 (Plot)	TS-V2 (Creativity)
Baseline	-	-	7.3 (1.76)	9.2 (1.94)	8.12 (2.24)	6.7 (2.10)	5.17 (1.68)
Transformer	24.78 (6.45)	2.89 (0.13)	6.81 (1.79)	8.87 (2.15)	7.48 (2.37)	6.07 (2.00)	4.80 (1.78)
ICLM	25.02 (6.89)	3.06 (0.14)	6.81 (1.75)	9.12 (1.86)	7.53 (2.44)	5.95 (2.14)	4.62 (1.72)

Table 1: Results on the GoodWiki [Gao et al. 2020] and TinyStories-V2 [Eldan and Li 2023, Nabeshima 2024] (TS-V2) datasets. GoodWiki models use $L = 12$ layers, 8 heads, and a 512-dimensional embedding, trained for 20 epochs. TS-V2 models use $L = 8$ layers, 8 heads, and a 512-dimensional embedding, trained for 10 epochs. The parentheses show standard deviation. *Baseline scores* reflect LLM evaluations of the quality of the original ground-truth dataset samples, serving as an approximate upper bound.

Model	# Param	SlimPajama 6B (PPL)	HellaSwag (ACC %)	WinoGrande (ACC %)	PIQA (ACC %)	ARC (ACC %)	ARC (Easy) (ACC %)
GPT-2 Medium	355M	-	37.8 (2.17)	53.0 (2.23)	68.0 (2.09)	19.2 (1.76)	48.2 (2.23)
Transformer	353M	17.90 (6.73)	37.0 (2.16)	52.2 (2.22)	64.4 (2.14)	19.6 (1.78)	44.8 (2.23)
ICLM	353M	18.57 (7.10)	36.0 (2.15)	49.6 (2.24)	64.0 (2.15)	20.6 (1.81)	44.8 (2.23)

Table 2: Models trained on SlimPajama-6B [Shen et al. 2024] for 6 epochs with $L = 24$ layers, 16 attention heads, and a 1024-dimensional embedding size. Zero-shot evaluation was performed using the LM Evaluation Harness (<https://github.com/EleutherAI/lm-evaluation-harness>), without any additional training or fine-tuning. GPT-2 Medium results [Wolf et al. 2019] are included for reference, though not directly comparable due to differing training pipelines and exact parameter count. The parentheses show standard deviation on perplexity and standard error on accuracy. Additional benchmarking details can be found in the SM.

behavior in the ϕ update, but here we find that the attention sinks *always* manifest within (at least) the start token, whereas the Transformer is less consistent with its location. This is similar to including register tokens [Burtsev et al. 2020, Darcet et al. 2023], but we note that our inclusion of the start token is a direct result of our underlying model theory, rather than a separate addition to the model. Perhaps even more interesting is that (in our initial studies) the attention sinks *only* appear within the ϕ update layers. The f update, on the other hand, shows significantly more balanced attention scores. This behavior is seen at all layers and is quite consistent between prompts. More samples are provided in the SM.

Future work is required to study this phenomenon in greater detail within ICLM and the Transformer, but by separating out the roles of ϕ and f , we expect this analysis will be easier within ICLM than by navigating the unrestricted attention within the Transformer (again, where different phenomena may intermingle).

In-Context Induction Behavior The structure of the f update layer bears a strong resemblance to the mechanism of induction heads [Olsson et al. 2022b, Nanda et al. 2023]. Induction heads are learned attention heads that update a Transformer’s prediction by attending to prior occurrences of a token in the context and effectively *copying* their representations. The f update imposes an even stricter inductive bias towards this behavior; its values solely come from the set of token embeddings of the context prior to the next token to be predicted (recall the w_{y_j} in (11)). This restriction makes it particularly interesting to study how such a bias shapes the model’s internal behaviors.

To study this, we developed a new dataset called **TinyMath**. Each text sample contains a short synthetically generated word problem that is similar in complexity and qual-

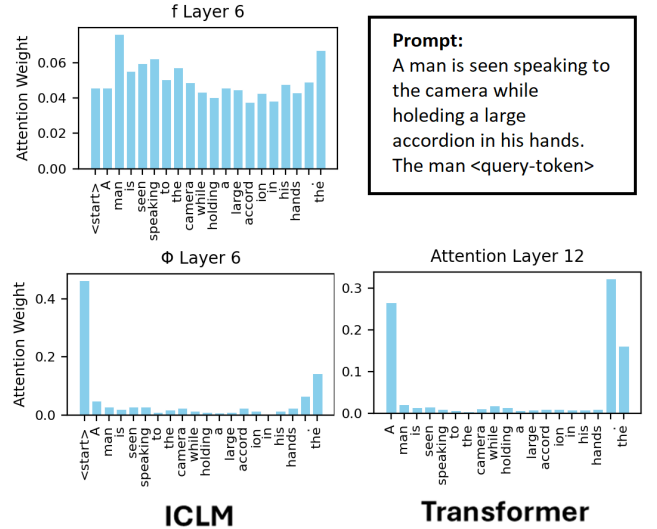


Figure 2: Attention heatmap for the medium size (353M parameter) ICLM and Transformer models. Layers shown are from halfway through the model (layer 12 for the Transformer and the 6th ϕ and f update for ICLM), but results were consistent across layers.

ity to TinyStories [Eldan and Li 2023], but contains a simple (two number) math problem. The TinyMath dataset is meant to examine the model’s capacity to perform simple reasoning, with such not studied with TinyStories data [Eldan and Li 2023]. The TinyMath stories were generated using GPT-4o-mini and GPT-4.1-nano models based on controlled equation selection and prompt seeding.

During *training* with TinyMath for ICLM or the Transformer, the explicit equation for the problem is also pro-

vided after the story, as well as a short written explanation of how to solve the problem and the final answer (like training with chain-of-thought reasoning [Yeo et al. 2025]). During testing, the models are tasked with generating the question, explanation, and answer, the latter used for evaluating accuracy. This dataset provides a controlled environment for analyzing both in-context and out-of-context token predictions; for a model to solve a problem correctly, it must first parse the context for the equation (including ignoring unrelated numbers and deciphering operator phrase patterns), and then rely on out-of-context memory to solve this equation. See Figure 3 for a sample prompt and generation, and Table 3 for performance results (discussed more later). TinyMath will be released upon publication of this paper, and additional details and performance studies can be found in the SM.

In Figure 3 we show attention heatmaps for models trained on the TinyMath dataset while generating the question from the story, a point at which they need to rely heavily on the context. The Transformer’s attention maps show no obvious patterns, outside of manifesting the previously mentioned attention sinks, and a more in-depth analysis of the attention heads will be needed to understand how the Transformer came to its final prediction.

ICLM shows a clear pattern: when predicting a token that appears within the context, the f attention score for that token dominates. This corresponds to placing a high weight on the (unmodified) embedding vector w_{y_j} for that token in the f update, which in ICLM corresponds directly to a high probability of predicting that token. Within the ϕ update itself this pattern is not seen, suggesting that the f update is more important to the manifestation of induction. This behavior is highly consistent, appearing at most layers and across all prompts during question generation. Additional figures are provided in the SM.

The simplicity of the models and dataset make it difficult to generalize this specific behavior to larger models, but they do suggest that the strong inductive biases and well structured nature of ICLM could serve as a cleaner testing ground for mechanistic analysis.

Model	Test PPL	Test ACC (%)
ICLM	1.76	63.90
ICLM (No MLP)	1.77	49.55
Transformer	1.74	64.50
Transformer (No MLP)	1.81	59.38

Table 3: Models with $L = 8$, trained on the TinyMath dataset. Perplexity and accuracy reported for a held-out test set, where the model was provided only the word problem and was evaluated on its final response and answer.

Analysis of MLP Layers

As discussed in Section 6, the MLP has been shown to play an important role in memory and reasoning skills [Dong et al. 2025, Geva et al. 2022, 2021], and we believe that its inclusion in the ICLM helps provide the heavily context-dependent model a similar out-of-context mechanism. Table 3, detailing model performance on TinyMath, shows that

Prompt:

Story: At 7 o’clock, Ben, age 9, arrived at the local library for a special event. Among 15 students, he proudly displayed his collection of **28** toy cars. As the evening went on, Ben learned about a charity event helping kids in need. Feeling generous, he decided to donate **12** of his cars to the cause. Everyone clapped and cheered for his kind heart. As he packed up, Ben wondered about something. What is the remaining number of cars?

Generated:

Question: 28 - **12** = ?

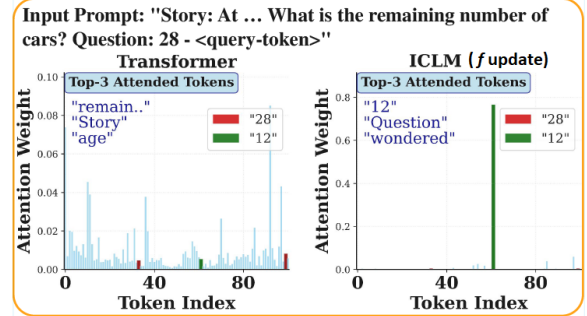


Figure 3: Attention maps for $L = 8$ Transformer and ICLM models trained on TinyMath. The prompt is shown at the top, with red and green tokens representing important in-context information and the orange box corresponding to the token being predicted. Additional samples can be found in the SM.

removing the MLP from either the ICLM or the Transformer model results in only a slight increase in perplexity, but a significant loss in accuracy for the math problem (which consists of addition, subtraction, multiplication or division of two numbers). This supports the idea the MLP in the ICLM plays a crucial role in its memory and reasoning capabilities, similar to the Transformer [Dong et al. 2025], while ICLM without the MLP remains effective at in-context modeling. We see consistent behavior in language-only tasks that require memory, with further results in the SM.

8 Conclusions

We have introduced ICLM, a novel language model that closely mirrors the Transformer while offering improved interpretability by separating feature extraction (ϕ) from prediction (f), and grounding the predictive process in functional GD. ICLM recovers many of the core behaviors observed in Transformers, while doing so in a way that is theoretically transparent and modular.

ICLM achieves competitive performance across a range of language modeling benchmarks while maintaining parity in parameter count. Its architecture also enables clear attribution of responsibility to different components, allowing isolation and study of phenomena such as attention sinks, kernel-based context matching, and the functional role of MLP layers in memory. Bridging mechanistic interpretability and theory, ICLM represents a new class of *model system* – a simplified, analyzable, yet expressive model that may serve as a faithful surrogate for more complex architectures, for mechanistic studies of language models.

References

- Ahn, K.; Cheng, X.; Daneshmand, H.; and Sra, S. 2023. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv:2306.00297*.
- Akyürek, E.; Andreas, J.; and Lin, K. 2022. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations (ICLR)*.
- Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.
- Burtsev, M. S.; Kuratov, Y.; Peganov, A.; and Sapunov, G. V. 2020. Memory Transformer. *arXiv preprint*, 2006.11527.
- Chen, Y.; Cao, P.; Chen, Y.; Liu, K.; and Zhao, J. 2025. Knowledge localization: Mission not accomplished? Enter query localization! *Int. Conf. Learning Representations (ICLR)*.
- Cheng, X.; Chen, Y.; and Sra, S. 2024. Transformers Implement Functional Gradient Descent to Learn Non-Linear Functions In Context. *arXiv:2312.06528*.
- Choi, E. 2023. GoodWiki Dataset. <https://www.github.com/euirim/goodwiki>.
- Choromanski, K.; Likhoshesterov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; Belanger, D.; Colwell, L.; and Weller, A. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations (ICLR)*.
- Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What does BERT look at? An analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP*, 276–286.
- Darcet, T.; Oquab, M.; Mairal, J.; and Bojanowski, P. 2023. Vision Transformers Need Registers. *CoRR*, abs/2309.16588.
- DeepSeek. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv:2501.12948v1*.
- DKYoon. ????. SlimPajama-6B dataset. <https://huggingface.co/datasets/DKYoon/SlimPajama-6B>. Accessed: 2025-08-02.
- Dong, Y.; Noci, L.; Khodak, M.; and Li, M. 2025. Attention Retrieves, MLP Memorizes: Disentangling Trainable Components in the Transformer. *arXiv:2506.01115v2*.
- Eldan, R.; and Li, Y. 2023. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? *arXiv:2305.07759*.
- Eldan, R.; Radford, A.; Brockman, G.; Amodei, D.; and Sutskever, I. 2023. TinyStories: How small can language models be and still speak coherent English? *arXiv preprint arXiv:2305.07759*.
- Elhage, N.; Nanda, N.; Olsson, C.; et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.
- Gao, L.; Biderman, S.; Black, S.; Golding, L.; Hoppe, T.; Foster, C.; Phang, J.; He, H.; Thite, A.; Nabeshima, N.; et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*.
- Garg, S.; Elhage, N.; Olsson, C.; Nanda, N.; Goldie, A.; Hernandez, D.; Joseph, N.; Mann, B.; Krueger, D.; and Amodei, D. 2022. Can transformers learn gradient descent? In *International Conference on Learning Representations*.
- Geva, M.; Caciularu, A.; Wang, K.; and Goldberg, Y. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *Proc. Conf. Empirical Methods in Natural Language Processing*.
- Geva, M.; Schuster, R.; Berant, J.; and Levy, O. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. *Proc. Conf. Empirical Methods in Natural Language Processing*.
- Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *Kernel Smoothing Methods*, 191–218. New York, NY: Springer New York.
- Kim, J.; Koo, B.; and Knoblich, J. 2020. Human organoids: model systems for human biology and medicine. *Nat. Rev. Mol. Cell Biol.*
- Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2016. Pointer Sentinel Mixture Models: beyond the softmax bottleneck.
- Nabeshima, N. 2024. TinyStoriesV2. <https://huggingface.co/datasets/noanabeshima/TinyStoriesV2>. Accessed: 2025-08-02.
- Nadaraya, E. A. 1964. On Estimating Regression. *Teoriya Veroyatnostei i Ee Primeneniya*.
- Nanda, N.; Lieberum, T.; Saulnier, L.; Belrose, G.; and Elhage, N. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *arXiv preprint arXiv:2301.05062*.
- Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Johnston, S.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2022a. In-context Learning and Induction Heads. *arXiv:2209.11895*.
- Olsson, C.; Ganguli, D.; Askell, A.; Henighan, T.; Hernandez, D.; Joseph, N.; Krueger, D.; Mann, B.; Nanda, N.; Schiefer, N.; et al. 2022b. In-context learning and induction heads. *Transformer Circuits Thread*.
- OpenAI. 2024. GPT-4o Technical Report. <https://openai.com/index/gpt-4o>. Accessed August 2025.
- OpenAI Community. ????. GPT-2 Medium model card. <https://huggingface.co/openai-community/gpt2-medium>. Accessed: 2025-08-02.
- Press, O.; and Wolf, L. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for*

Computational Linguistics: Volume 2, Short Papers, 157–163. Association for Computational Linguistics.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. OpenAI.

Schölkopf, B.; and Smola, A. 2002. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Shen, Z.; Tao, T.; Ma, L.; Neiswanger, W.; Liu, Z.; Wang, H.; Tan, B.; Hestness, J.; Vassiliev, N.; Soboleva, D.; and Xing, E. 2024. SlimPajama-DC: Understanding Data Combinations for LLM Training. arXiv:2309.10818.

Su, J.; Lu, Y.; Pan, S.; Murtadha, A.; Wen, B.; and Liu, Y. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864.

Su, J.; Lu, Y.; Pan, S.; Wen, B.; and Liu, Y. 2021. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv preprint arXiv:2104.09864.

Tsai, Y.-H. H.; Bai, S.; Yamada, M.; Morency, L.-P.; and Salakhutdinov, R. 2019. Transformer Dissection: A Unified Understanding for Transformer’s Attention via the Lens of Kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4354–4363.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vig, J.; Belinkov, Y.; Neves, L.; Saphra, N.; and Bowman, S. R. 2020. Investigating multilingual NLP models’ zero-shot capabilities: A case study on the XGLUE benchmark. arXiv preprint arXiv:2008.04513.

von Oswald, J.; Niklasson, E.; Randazzo, E.; Sacramento, J.; Mordvintsev, A.; Zhmoginov, A.; and Vladymyrov, M. 2023. Transformers learn in-context by gradient descent. arXiv:2212.07677.

Wang, A.; Convertino, W.; Cheng, X.; Henao, R.; and Carin, L. 2025. On Understanding Attention-Based In-Context Learning for Categorical Data. *Int. Conf. Machine Learning (ICML)*.

Watson, G. S. 1964. Smooth Regression Analysis. *Sankhyā, Series A*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; et al. 2019. gpt2-medium · Hugging Face. <https://huggingface.co/gpt2-medium>. Accessed: 2025-08-02.

Xiao, G.; Tian, Y.; Chen, B.; Han, S.; and Lewis, M. 2024. Efficient Streaming Language Models with Attention Sinks. *Int. Conf. Learning Representations (ICLR)*.

Yeo, E.; Tong, Y.; Niu, M.; Neubig, G.; and Yue, X. 2025. Demystifying Long Chain-of-Thought Reasoning in LLMs. arXiv:2502.03373.

Reproducibility Checklist

This paper:

Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA)
Yes, in Figures 1 and 2 we provide conceptual outlines.

Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no)
Yes

Provides well marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no)
Yes

Does this paper make theoretical contributions? (yes/no)
Yes

If yes, please complete the list below.

All assumptions and restrictions are stated clearly and formally. (yes/partial/no)
Yes

All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no)
Yes

Proofs of all novel claims are included. (yes/partial/no)
Yes (in Appendix we derive in detail all underlying math connected to our claims)

Proof sketches or intuitions are given for complex and/or novel results. (yes/partial/no)
Yes

Appropriate citations to theoretical tools used are given. (yes/partial/no)
Yes

All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA)
Yes

All experimental code used to eliminate or disprove claims is included. (yes/no/NA)
Yes

Does this paper rely on one or more datasets? (yes/no)
Yes

If yes, please complete the list below.

A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA)
Yes

All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA)

Yes

All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no/NA)

Yes

All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes/no/NA)

Yes

All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes/partial/no/NA)

Yes

All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisfying. (yes/partial/no/NA)

NA

Does this paper include computational experiments? (yes/no)

Yes

If yes, please complete the list below.

This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes/partial/no/NA)

Yes

Any code required for pre-processing data is included in the appendix. (yes/partial/no).

Yes

All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no)

Yes

All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no)

Yes

All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no)

Yes

If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA)

NA

This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes/partial/no)

Yes

This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes/partial/no)

Yes

This paper states the number of algorithm runs used to compute each reported result. (yes/no)

Yes

Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes/no)

Yes

The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no)

Yes

This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no/NA)

Yes

A Derivation of the GD Update Equation

First consider $f(\phi) = A\psi(\phi)$, the cost function for inferring the parameters A , with $A \in \mathbb{R}^{d' \times d_\psi}$, may be expressed as

$$\begin{aligned}\mathcal{L}(A) &= -\frac{1}{N} \sum_{i=1}^N \log \left[\frac{\exp[w_{y_i}^\top A\psi(\phi_i)]}{\sum_{c=1}^C \exp[w_c^\top A\psi(\phi_i)]} \right] \\ &= -\frac{1}{N} \sum_{i=1}^N [w_{y_i}^\top A\psi(\phi_i) - \log \sum_{c=1}^C \exp(w_c^\top A\psi(\phi_i))] .\end{aligned}\tag{13}$$

let a_m represent the m th row of A . Taking the gradient of \mathcal{L} wrt a_m :

$$\begin{aligned}\nabla_{a_m} \mathcal{L} &= -\frac{1}{N} \sum_{i=1}^N \left[w_{y_i}(m)\psi(\phi_i) - \frac{\sum_{c=1}^C \exp[w_c^\top A\psi(\phi_i)] w_c(m)\psi(\phi_i)}{\sum_{c'=1}^C \exp(w_{c'}^\top A\psi(\phi_i))} \right] \\ &= -\frac{1}{N} \sum_{i=1}^N \left[w_{y_i}(m) - \frac{\sum_{c=1}^C \exp[w_c^\top f_i] w_c(m)}{\sum_{c'=1}^C \exp(w_{c'}^\top f_i)} \right] \psi(\phi_i) \\ &= -\frac{1}{N} \sum_{i=1}^N [w_{y_i}(m) - \mathbb{E}(w(m)|f_i)] \psi(\phi_i) .\end{aligned}\tag{14}$$

where $w_{y_i}(m)$ is the m th component of the embedding vector w_{y_i} for token y_i and

$$\mathbb{E}(w(m)|f_i) = \sum_{c=1}^C w_c(m) \left[\frac{\exp[w_c^\top f_i]}{\sum_{c'=1}^C \exp(w_{c'}^\top f_i)} \right]\tag{15}$$

represents the expected value of component j of the embedding vectors, based on a softmax probability over categories (tokens) using $f_i = f(\phi_i)$.

The gradient update step (from step k to $k+1$) for a_m is

$$\begin{aligned}a_{m,k+1} &= a_{m,k} - \alpha \nabla_{a_m} \mathcal{L} \\ &= a_{m,k} + \frac{\alpha}{N} \sum_{i=1}^N [w_{y_i}(m) - \mathbb{E}(w(m)|f_i)] \psi(\phi_i) .\end{aligned}$$

Using the GD update rules for $\{a_m\}_{j=1,d'}$, we have

$$\begin{aligned}f_{j,\ell+1} &= \begin{pmatrix} a_{1,\ell+1}^\top \psi(\phi_j) \\ \vdots \\ a_{d',\ell+1}^\top \psi(\phi_j) \end{pmatrix} \\ &= f_{j,\ell} + \frac{1}{N} \Lambda \sum_{i=1}^N [w_{y_i} - \mathbb{E}(w|f_{i,\ell})] \kappa(\phi_i, \phi_j)\end{aligned}\tag{16}$$

where $\Lambda = \text{diag}(\alpha_1, \dots, \alpha_{d'})$, assuming a different learning rate for each of the d' components of $f(\phi)$. The expression $f_{j,\ell} = f_\ell(\phi_j)$ represents the latent function at covariates ϕ_j , for GD step k .

If this is generalized to H kernels, and hence $f(\phi) = \sum_{h=1}^H A_h \psi_h(\phi)$, then the above analysis applied to each of the A_h yields

$$f_{j,\ell+1} = f_{j,\ell} + \frac{1}{N} \sum_{h=1}^H \Lambda_h \sum_{i=1}^N [w_{y_i} - \mathbb{E}(w|f_{i,k})] \kappa_h(\phi_i, \phi_j)\tag{17}$$

where $\kappa_h(\phi_i, \phi_j) = \psi_h^\top(\phi_i) \psi_h(\phi_j)$.

In the above analysis, we have assumed that each Λ_h was a diagonal matrix. However, extending this concept to *preconditioned* functional GD [Ahn et al. 2023], each diagonal Λ_h may be replaced with a (learned) full matrix, yielding (7). We do not use the $1/N$ in practice, as the kernel attention is replaced with softmax attention, as discussed in Section B.

B From RKHS to Softmax Attention

To simplify notation, in (5) we omitted a factor $1/N$ that normalizes the cross-entropy loss by context length. While this normalization is irrelevant for fixed-length contexts, our framework compares contextual sequences of variable length. Including this factor leads to a scaling term $1/(i-1)$ in front of the sums in (11), which we omitted earlier to simplify exposition. While such corrections can be incorporated in principle, we instead adopt a different and more general approach grounded in kernel smoothing.

Although (11) was derived within an RKHS framework with symmetric, unnormalized kernels, it is natural to interpret the update as a form of kernel regression. In particular, we may normalize the kernel weights using Nadaraya-Watson averaging [Nadaraya 1964, Watson 1964, Hastie, Tibshirani, and Friedman 2009], yielding:

$$\kappa(W_{K,h}^{(\ell)}\phi_{i,\ell}, W_{Q,h}^{(\ell)}\phi_{j,\ell}) \rightarrow \frac{\kappa(W_{K,h}^{(\ell)}\phi_{i,\ell}, W_{Q,h}^{(\ell)}\phi_{j,\ell})}{\sum_{i=1}^{j-1} \kappa(W_{K,h}^{(\ell)}\phi_{i,\ell}, W_{Q,h}^{(\ell)}\phi_{j,\ell})} \quad (18)$$

A particularly important special case is when κ is the exponential dot-product kernel:

$$\kappa(W_{K,h}^{(\ell)}\phi_{i,\ell}, W_{Q,h}^{(\ell)}\phi_{j,\ell}) = \exp \left[\lambda (W_{K,h}^{(\ell)}\phi_{i,\ell})^\top W_{Q,h}^{(\ell)}\phi_{j,\ell} \right],$$

which corresponds exactly to softmax-based attention as used in conventional Transformers [Vaswani et al. 2017].

Replacing the symmetric RKHS kernel with softmax attention not only simplifies normalization but aligns our functional GD framework with standard Transformer practice. Prior work has shown that dot-product attention functions as a learned, context-adaptive kernel over feature representations [Tsai et al. 2019, Choromanski et al. 2021]. Though softmax is not a classical RKHS kernel, it behaves analogously in practice. It models similarity between context vectors and defines attention weights, effectively functioning as a learned adaptive kernel.