INTERPRETING MULTI-LAYER TRANSFORMERS FOR IN-CONTEXT LINEAR REGRESSION WITH VARYING COVARIANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how multi-layer softmax-attention transformers perform in-context linear regression in the challenging regime where the covariate distribution varies across sequences, in contrast to the fixed-distribution setting considered in prior work. We first show that conclusions from prior work break down in this regime, and that transformer depth becomes essential for achieving low in-context error. Through a population-limit interpretation of the trained transformer, we reveal that it solves linear regression by implementing a variant of gradient descent whose parameters depend strongly on model depth, corresponding to a consistent structure in the learned weight matrices. We also discover the non-convergent property of the learned algorithm. Building on this insight, we show that with a chain-of-thought-style intermediate step, the transformer can solve in-context instrumental variable (IV) regression, applying the same algorithm whenever the same regime studied here appears in the IV regression. Our findings provide new evidence that multi-layer transformers learn distinct in-context algorithms under more complex regression scenarios, bridging the gap between empirical performance and interpretable algorithmic behavior.

1 INTRODUCTION

Transformer-based Large Language Models (LLMs) (Touvron et al., 2023a;b) have transformed artificial intelligence, in no small part due to their capacity for in-context learning (ICL) (Bhattamishra et al., 2023; Park et al., 2024; Tang et al., 2023), the ability to solve novel tasks from a handful of demonstrations in the prompt without any parameter updates. Understanding mechanisms behind this emergent capability has become a central question in the field (Brown et al., 2020; Dong et al., 2022).

A productive line of work approaches this question through controlled synthetic settings and aims at a precise, mathematically rigorous interpretation of the algorithms learned by transformers, most notably in-context linear regression (Garg et al., 2022). In the most common setup, a transformer is trained on sequences of input–output pairs $\{(x_i, y_i)\}_{i=1}^n$ with the implicit assumption that inputs drawn from a fixed Gaussian distribution $\mathcal{N}(0, \Sigma)$ (Ahn et al., 2023; Zhang et al., 2024; Akyürek et al., 2022). Under this assumption, the transformer can encode the covariance matrix Σ into its weights as an inductive bias, and a single attention layer suffices to implement one step of preconditioned Gradient Descent (GD) (Ahn et al., 2023; He et al., 2025) to solve in-context linear regression tasks. While elegant, this mechanism faces to a problem: once the input covariance varies across sequences, such models fail to generalize (Von Oswald et al., 2023).

In this paper, we study the substantially more realistic and challenging regime in which the covariance matrix Σ varies across sequences. This regime arises naturally in downstream tasks such as instrumental variable (IV) regression (Liang et al., 2024). Crucially, the test-time input distribution need not be seen during training, so the model cannot simply bake a single Σ into its weights. In this regime, we empirically observe that the single-layer transformer, which is sufficient in prior work, incurs a performance gap compared to one-step GD, indicating that existing conclusions no longer apply. This motivates two fundamental questions:

(1) Can a transformer be trained to solve linear regression in context with varying covariances?
(2) If so, what linear regression algorithm does it learn to implement?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To address the first question, we train looped multi-layer softmax-attention transformers (Giannou et al., 2023; Fan et al., 2024), which is currently adopted in many scenarios (Geiping et al., 2026; Zhu et al., 2025), under the varying-covariance regime. We find that success depends critically on architectural choices. In particular, ICL performance improves monotonically with depth, with the trend saturating beyond roughly three layers. Sufficient depth is therefore a necessary ingredient for learning this harder task.

To address the second question, we first identify a consistent diagonal structure in the attention circuits, and then interpret the trained weight matrices in the population limit, finding that the transformer implements a novel variant of the GD algorithm, composed of two operations: (1) a linear transformation of the covariates x_i; and (2) a GD-like update on a reformulated regression problem that uses these transformed covariates. The algorithm is adaptive to the various covariance of the input sequences. Strikingly, unlike standard multi-step GD, the learned procedure is non-convergent: the loss grows rapidly past an optimal number of iterations. This suggests that the learned step sizes are thus optimized for the exact transformer depth.

Building on these findings, we apply our analysis to in-context IV regression, whose classical solution is the two-stage least squares (2SLS) estimator. In the first stage, 2SLS regresses the endogenous covariate x_i on an instrument z_i to obtain fitted values $\hat{x}_i$; in the second stage, it regresses the outcome y_i on $\hat{x}_i$. In the in-context setting, the distribution of $\hat{x}_i$ varies across sequences, which matches our regime. We design a multi-layer transformer with a Chain-of-Thought (CoT) (Wei et al., 2022) structure that mirrors this decomposition: a first block predicts $\hat{x}_i$, and a deeper block regresses y_i on $\hat{x}_i$, with both blocks trained jointly. Empirically, the first block learns a pre-conditioned GD estimator, while the second block learns the same algorithm identified in our linear regression regime. The resulting model attains performance comparable to the classical 2SLS estimator.

Overall, our main contributions are as follows:

- **Empirical analysis under varying covariance.** We analyze on a novel linear regression regime, where the single-layer construction of prior work fails, and looped multi-layer transformers succeed with ICL loss saturating at three layers and a consistent diagonal structure emerging in the learned KQ/OV circuits.
- **Theoretical interpretation of the learned algorithm.** Via a population-limit analysis, we identify the algorithm the transformer implements as an iterative variant of gradient descent whose step sizes are *tuned to the model depth*. The interpreted algorithm matches the trained transformer in both prediction loss and input sensitivity, and explains the model's divergent behavior under depth extrapolation.
- **Extension to in-context IV regression.** Building on this interpretation, we design a chain-of-thought-style two-block transformer for instrumental-variable regression and show that, empirically, it tracks the two-stage least squares estimator. The second block reuses the same algorithm identified above, demonstrating that the interpretation transfers to a non-trivial downstream task.

2 Related Work

In-Context Learning. Large language models (LLMs) (Floridi & Chiriatti, 2020; Achiam et al., 2023) owe much of their empirical success to in-context learning (ICL), the ability to adapt to new tasks at inference time using only a prompt of input–output examples, without any parameter updates (Liu et al., 2021; Min et al., 2021; Nie et al., 2022; Bai et al., 2023). Despite this progress, a unified theoretical understanding of ICL remains an active area of research.

Existing theoretical work has taken several approaches. One prominent line interprets ICL as an implicit form of Bayesian inference implemented by the transformer's attention mechanism (Zhang et al., 2023; Jeon et al., 2024; Hu et al., 2024). Another analyzes how transformers can simulate learning algorithms, such as gradient descent or kernel methods, to solve ICL tasks (Bai et al., 2023; Zhang et al., 2025; Nichani et al., 2024; Sheen et al., 2024; Giannou et al., 2024). Other studies investigate the function classes that transformers can represent or learn in context (Garg et al., 2022), or explore emergent behaviors and the transient nature of ICL capabilities (Singh et al., 2023; Dai et al., 2022; Olsson et al., 2022). Our work contributes to this area by focusing on how multi-layer softmax attention theoretically and empirically enables in-context linear regression.

In-Context Linear and IV Regression. To better understand how transformers acquire ICL abilities, researchers have extensively studied simple, interpretable tasks such as linear regression. The pioneering work (Garg et al., 2022) empirically showed that transformers can achieve near-Bayes-optimal performance on these tasks. Subsequent theoretical analyses (Ahn et al., 2023; Zhang et al., 2024; Von Oswald et al., 2023; Akyürek et al., 2022; Mahankali et al., 2023) revealed that linear transformers can implement optimization algorithms, such as gradient descent and its variants. Further studies have focused on more complex attention mechanisms with non-linear activations (He et al., 2025; Chen et al., 2024; Cheng et al., 2023; Cui et al., 2024; Fu et al., 2024). In addition, several works (Li et al.; 2025; Oh et al., 2025) have studied in-context learning in gated linear attention models (Gu & Dao, 2024; Yang et al.). Unlike most existing works that assume a fixed covariate distribution across sequences, we focus on the more challenging regime where the data distribution varies across sequences. We also extend our findings to more complex tasks such as IV regression, which has been investigated in other recent studies (Liang et al., 2024; Wang et al., 2024).

3 Preliminary

In this study, we largely follow prior experimental setups (Garg et al., 2022; He et al., 2025), with the key difference that we allow the covariate distribution to vary across sequences.

3.1 Data Distribution and Embedding

The linear regression task involves L examples $\{(x_\ell, y_\ell)\}_{\ell\in[L]}$, where $x_\ell \in \mathbb{R}^d$ and $y_\ell \in \mathbb{R}$. The covariates x_ℓ in each sequence are drawn i.i.d. from $\mathsf{P}_x = \mathcal{N}(0, \Sigma)$. In contrast to previous studies, we consider a more challenging scenario in which Σ varies across sequences during training. We randomly generate different Σ for different sequences. Specifically, we set $\Sigma = U\mathrm{diag}(\lambda_1, \ldots, \lambda_d)U^\top$, where U is a rotation matrix obtaining from the QR decomposition of a random matrix M. The matrix M is different across different sequences. The eigenvalues λ_i are sampled i.i.d. from $\mathrm{Unif}(\lambda_{\min}, \lambda_{\max})$. The label y_ℓ for each x_ℓ is given by $y_\ell = \beta^\top x_\ell + \epsilon_\ell$, where the true regression vector $\beta \in \mathbb{R}^d$ is drawn from $\mathsf{P}_\beta = \mathcal{N}(0, I_d)$ and the noise term ϵ_ℓ is sampled from $\mathcal{N}(0, \sigma^2)$. A test input x_q is independently drawn from a distribution that may differ from the training distribution $\mathcal{N}(0, \Sigma)$. To facilitate in-context learning, we construct an embedding matrix $Z^{(0)}_{\texttt{ebd}} \in \mathbb{R}^{(d+1)\times(L+1)}$ by concatenating the L training pairs $\{(x_\ell, y_\ell)\}_{\ell\in[L]}$ with the test input x_q and a placeholder zero:

$$Z^{(0)}_{\texttt{ebd}} = [Z \;\; z_q] = \begin{bmatrix} x_1 & x_2 & \ldots & x_\ell & x_q \\ y_1 & y_2 & \ldots & y_\ell & 0 \end{bmatrix}. \tag{1}$$

3.2 Transformer Architecture

Multi-Head Softmax Attention. We consider sequence-to-sequence multi-head softmax attention with H parallel heads and parameters $\theta = \{O^{(h)}, V^{(h)}, K^{(h)}, Q^{(h)}\}_{h\in[H]} \subseteq \mathbb{R}^{(d+1)\times(d+1)}$. The attention operation is given by

$$\texttt{Attn}_\theta(Z_{\texttt{ebd}}) = \sum_{h=1}^{H} O^{(h)}V^{(h)}Z_{\texttt{ebd}} \cdot \texttt{smax} \circ \texttt{msk}(Z_{\texttt{ebd}}^\top {K^{(h)}}^\top Q^{(h)} Z_{\texttt{ebd}}), \tag{2}$$

where $\texttt{smax}(\cdot)$ denotes the column-wise softmax operation and $\texttt{msk}(\cdot)$ applies a prefix mask excluding the final query token from the key-value set (He et al., 2025), i.e., $\texttt{msk}(S_{ij}) = S_{ij}$ for $i \le L$ and $\texttt{msk}(S_{L+1,j}) = -\infty$ for every j. This enables every token to attend to the same L in-context keys.

We primarily use looped multi-layer transformers with residual connections in our study. For a transformer with T layers, the output $Z^{(t)}_{\texttt{ebd}}$ at layer t is given by

$$Z^{(t)}_{\texttt{ebd}} = \texttt{TF}_\theta(Z^{(t-1)}_{\texttt{ebd}}) = Z^{(t-1)}_{\texttt{ebd}} + \texttt{Attn}_\theta(Z^{(t-1)}_{\texttt{ebd}}), \tag{3}$$

where $Z^{(t-1)}_{\texttt{ebd}}$ denotes the input to layer t (i.e., the output of layer $t-1$). From the transformer's final output matrix $Z^{(T)}_{\texttt{ebd}}$, we extract the $(d+1, L+1)$-th entry as the prediction $\hat{y}_q \in \mathbb{R}$. The goal of $\hat{y}_q$ is to estimate $\mathbb{E}[y_q \mid x_q] = \beta^\top x_q$.

KQ and OV Circuits. The computation in Equation equation 2 involves two components: computing the similarity between queries and keys, and using the outputs of the softmax function to generate the layer output. These computations depend on the matrix products $K^{(t,h)^\top} Q^{(t,h)}$ and $O^{(t,h)} V^{(t,h)}$, respectively, where (t, h) refers to the t-th layer and h-th head. To simplify the notation, we denote $K^{(t,h)^\top} Q^{(t,h)}$ by $\Phi^{(t,h)} \in \mathbb{R}^{(d+1)\times(d+1)}$ and $O^{(t,h)} V^{(t,h)}$ by $\Psi^{(t,h)} \in \mathbb{R}^{(d+1)\times(d+1)}$. Because we use a looped transformer, $\Phi^{(t,h)}$ and $\Psi^{(t,h)}$ are shared across layers t.

Training Objective. We train the model by minimizing the mean squared error,

$$\mathcal{L}(\theta) = \mathbb{E}[(y_q - \hat{y}_q)^2] = \mathbb{E}[(y_q - \mathtt{TF}_\theta(Z^{(0)}_{\mathtt{ebd}}))^2]. \tag{4}$$

Optimization proceeds via a mini-batch gradient-based algorithm (Adam (Kingma & Ba, 2014) in practice), with $\nabla\mathcal{L}(\theta)$ computed for every projection matrix in the parameter set θ.

4 Mechanistic Interpretation of Multi-Layer Transformer

The most related prior works (Von Oswald et al., 2023; He et al., 2025) largely focus on single-layer models trained under a fixed input distribution. Such assumptions limit our understanding of how transformers generalize to covariates with different distributions. Here, we systematically examine multi-layer transformers trained on covariates with diverse covariance structures across sequences. We analyze their patterns and outputs, and interpret the algorithm learned by the looped transformer.

4.1 Empirical Results

Experimental Setup. For most experiments in the paper, we set the batch size to $n_{\text{batch}} = 256$, the dimension of x_ℓ to $d = 5$, the sequence length to $L = 200$, the noise variance to $\sigma^2 = 0.1$, the minimum eigenvalue to $\lambda_{\min} = 0.8$, and the maximum eigenvalue to $\lambda_{\max} = 1.2$. We train the 2-, 3-, and 4-layer looped transformers with different numbers of heads for 100,000, 150,000, and 200,000 iterations, respectively. All weight matrices are initialized with small random values and are optimized using the Adam optimizer with a learning rate of 10^{-3}.

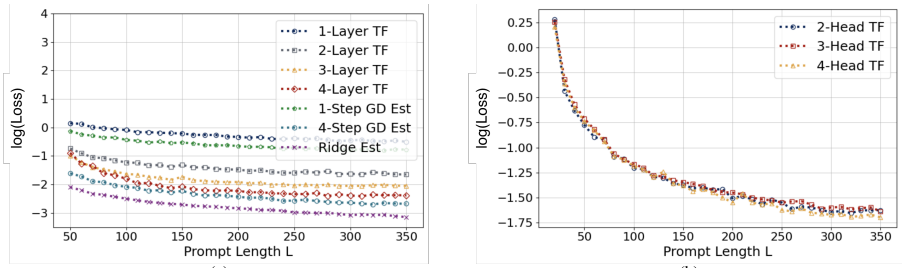


Figure 1: ICL prediction errors among: (a) transformer models and canonical estimators; (b) transformers with different heads.

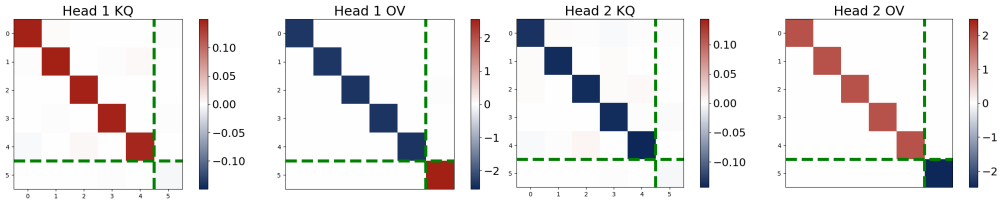


Figure 2: The learned patterns of 3-layer looped Transformer model.

Observation 1. Under our regime, the ICL loss of the single-layer transformer has a distinct gap compared to the one-step GD with optimal learning rate, which is different from that reported under the fixed-distribution regime in prior work (He et al., 2025). Moreover, transformer performance improves with increasing depth, and the gains become negligible once $T \geq 3$.

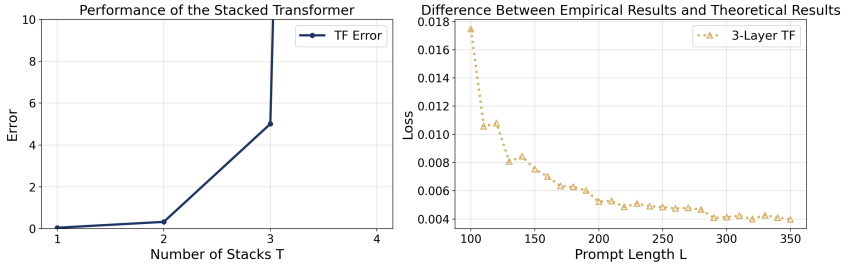


Figure 3: Performance about: (a) trained transformer different unrolling times; (b) error between population-limit theoretical values and empirical values.

Figure 1 reports the ICL loss across transformer depths at inference time. As shown in the figure, the multi-layer transformers have comparable performance (from around 0.03 to 0.01) compared to the 4-step GD and ridge estimator. Here the learning rate of the GD is optimally searched based on the data. The single-layer model performs poorly compared to the 4-step GD and ridge estimator (at around 0.3) under heterogeneous covariance, along with an around 0.15 gap compared to the one-step GD. This contrasts sharply with the prior conclusion that a single-layer transformer learns an estimator that fits the in-context linear regression tasks well with a fixed input distribution. Adding layers steadily improves performance, but the benefit saturates: beyond three layers, additional depth yields almost no gain, with the 3- and 4-layer curves nearly indistinguishable (at around 0.01).

Observation 2. For all multi-layer transformer models, the Key-Query (KQ) and Output-Value (OV) circuits exhibit a consistent structure across layers and heads, of the form

$$\Phi^{(h)} = \begin{bmatrix} \omega^{(h)} I_d & 0_d \\ 0_d^\top & 0 \end{bmatrix}, \qquad \Psi^{(h)} = \begin{bmatrix} u_1^{(h)} I_d & 0_d \\ 0_d^\top & u_2^{(h)} \end{bmatrix}, \tag{5}$$

where I_d denotes the $d \times d$ identity matrix. Because we employ a looped transformer, the KQ and OV circuits are shared across layers, so we omit the layer superscript t.

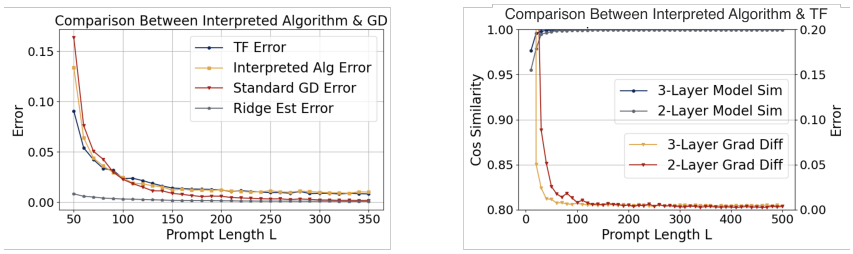
Figure 2 illustrates the KQ and OV circuit patterns for the three-layer transformer with two attention heads. Each diagonal entries of the learned KQ circuit are approximately constant, denoted by some $\omega^{(h)} \in \mathbb{R}$, with the exception of the last entry, which is zero. Similarly, in the OV circuit, the first d diagonal entries are approximately equal to $u_1^{(h)} \in \mathbb{R}$, while the last entry is equal to $u_2^{(h)} \in \mathbb{R}$. In addition, the same circuit patterns can also be seen across the different numbers of layers (shown in Figure 7 in appendix), irrespective of the number of layers. This suggests that, under our regime, the transformer learns a new algorithm that differs from those identified in prior work (He et al., 2025; Von Oswald et al., 2023), and newly learned algorithm is independent of any specific covariance structure of the training sequences.

Observation 3. The trained transformer generalizes well across sequence lengths. However, when a well-trained transformer block is unrolled more than once during inference, the ICL prediction loss does not decrease; in fact, it increases dramatically.

Figure 3 (a) illustrates this observation of a 3-layer looped transformer. This suggests that multi-layer transformers implement a divergent iterative procedure, with each model optimized only for its own depth, in contrast to standard multi-step GD.

Observation 4. For $H \geq 2$, transformers of the same depth achieve similar performance regardless of the number of heads.

Figure 1(b) shows that all multi-head models achieve similar ICL errors, suggesting that multi-head transformers with $H \geq 2$ are essentially equivalent to the two-head baseline. The single-head case is an exception, and we discuss it in Appendix B.1, which also contains additional visualizations and experimental results with larger input dimensions and the border range of the eigenvalue λ, demonstrating the generalization of our observation.



(a) (b)

Figure 4: (a) The performance comparison among 3-layer looped transformer, its interpreted algorithm, and 3-step GD; (b) the alignment between transformer and interpreted algorithm.

4.2 Insights into the Patterns in the Transformer

As shown in Section 4.1, trained transformers display clear structural patterns in their KQ and OV circuits, and we can interpret the algorithm learned by the transformer.

We begin with a general interpretation of the output of each attention layer from a Gaussian perspective. Note that this interpretation applies to a general transformer, independent of whether the weights are shared across layers. Accordingly, we retain the superscript t to denote layer-specific parameters. For analytical tractability, we study the *population-limit* situation, in which the empirical softmax average over the L in-context tokens is replaced by its expectation under the data distribution, *i.e.,*the $L \to \infty$ idealization of finite-L setting.

Theorem 1 (Population-Limit Layer-wise Dynamics)**.** *Consider a T-layer, H-head softmax attention transformer with KQ circuits $\Phi^{(t,h)}$ and OV circuits $\Psi^{(t,h)}$ at layer $t \in \{1, \dots, T\}$ and head $h \in [H]$, equipped with the prefix mask of Section 3 (every token attends to the same L in-context keys). We make the following three assumptions:*

(i) In-context tokens are i.i.d. Gaussian*: $\{z_\ell^{(0)}\}_{\ell \in [L]} \overset{i.i.d.}{\sim} p_0 := \mathcal{N}(\mu, \Sigma)$ with $\Sigma \succ 0$;*

(ii) Query token is arbitrary*: $z_q^{(0)} \in \mathbb{R}^{d+1}$ is fixed (or drawn from any distribution, possibly with degenerate covariance);*

(iii) Layer-wise non-degeneracy*: for every $t \in \{0, \dots, T-1\}$, the matrix*

$$A^{(t)} := I_{d+1} + \sum_{h=1}^{H} \Psi^{(t+1,h)} \Sigma^{(t)} \Phi^{(t+1,h)}$$

is non-singular, where $\Sigma^{(t)}$ is propagated by the recursion below.

Then, in the population limit, for every $t \in \{0, 1, \dots, T-1\}$ and every token index $\ell \in [L] \cup \{q\}$ including the query, the layer-wise output satisfies the exact affine recursion

$$z_\ell^{(t+1)} = A^{(t)} z_\ell^{(t)} + b^{(t)}, \qquad b^{(t)} := \sum_{h=1}^{H} \Psi^{(t+1,h)} \mu^{(t)}, \tag{6}$$

where the in-context Gaussian parameters evolve as

$$\mu^{(t+1)} = A^{(t)} \mu^{(t)} + b^{(t)}, \qquad \Sigma^{(t+1)} = A^{(t)} \Sigma^{(t)} \big(A^{(t)}\big)^\top, \tag{7}$$

with $\mu^{(0)} = \mu$ and $\Sigma^{(0)} = \Sigma$. By Assumption (iii), $\Sigma^{(t)} \succ 0$ for every t, so $p_t := \mathcal{N}(\mu^{(t)}, \Sigma^{(t)})$ is well-defined and Gaussian.

Theorem 1 offers an operator-level perspective on a multi-head attention layer: in the population limit, the t-th layer is an exact affine map taking the Gaussian distribution $\mathcal{N}(\mu^{(t-1)}, \Sigma^{(t-1)})$ over in-context tokens to $\mathcal{N}(\mu^{(t)}, \Sigma^{(t)})$, parameterized entirely by $\{\Psi^{(t,h)}, \Phi^{(t,h)}\}_{h \in [H]}$, and the same affine map carries the query token through layer-by-layer regardless of its marginal distribution. The full induction is given in Appendix A.1.

The population limit is the $L \to \infty$ idealization in which the empirical softmax average over the L in-context tokens is replaced by its expectation under p_t. At finite L, the trajectory differs from the population-limit trajectory by an additive residual of order $O_p(L^{-1/2})$ per layer, by standard concentration of the empirical softmax around its expectation; we treat this gap as a modeling approximation rather than fold it into the formal statement. Figure 3(b) compares empirical transformer outputs against the corresponding population-limit theoretical values as L increases, and the strong agreement validates both our recursive interpretation *and* the population approximation at our operating L.

The regime studied in the main paper is the special case of Equation equation 6 in which $\mu = 0$ and Σ varies across sequences.

Given the zero-mean inputs z_ℓ, the transformer's output form, and the learned structural patterns in Equation equation 5, we obtain the following Corollary 1. Because our transformer is looped, the KQ and OV parameters $\{u_1^{(t)}, u_2^{(t)}, \omega^{(t)}\}_{t=1}^T$ are the same at each t-th layer, so we drop the superscript t from (u_1, u_2, ω) hereafter.

Corollary 1 (Layer-wise Update under Structural Patterns)**.** *Under the assumptions of Theorem 1 and the learned circuit patterns in Equation equation 5, the layer-wise iteration* $(x_\ell^{(t)}, y_\ell^{(t)}) \mapsto (x_\ell^{(t+1)}, y_\ell^{(t+1)})$ *satisfies, for all* $\ell \in [L] \cup \{q\}$,

$$x_\ell^{(t+1)} = x_\ell^{(t)} + \frac{\langle u_1, \omega \rangle}{L} \sum_{\ell'=1}^{L} x_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)}, \qquad y_\ell^{(t+1)} = y_\ell^{(t)} + \frac{\langle u_2, \omega \rangle}{L} \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)},$$

where $\omega = (\omega^{(1)}, \ldots, \omega^{(H)}) \in \mathbb{R}^H$, $u_1 = (u_1^{(1)}, \ldots, u_1^{(H)}) \in \mathbb{R}^H$, *and* $u_2 = (u_2^{(1)}, \ldots, u_2^{(H)}) \in \mathbb{R}^H$. $\langle \cdot, \cdot \rangle$ *represents the inner product.*

This update rule can be summarized as the iterative procedure given in Algorithm 1. The detailed construction is provided in Appendix A.2. Hereafter, we refer this as the interpreted algorithm.

Algorithm 1 Interpreted Algorithm Implemented by Multi-Layer Transformer

Require: Step-sizes $\eta_x = \langle u_1, \omega \rangle$, $\eta_y = \langle u_2, \omega \rangle$, iterations time T

1: **for** each $t \in \{0, 1, \ldots, T-1\}$ and $\ell \in [L] \cup \{q\}$ **do**

2: $\quad x_\ell^{(t+1)} \leftarrow x_\ell^{(t)} + \eta_x / L \cdot \sum_{\ell'=1}^{L} x_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)}$

3: $\quad y_\ell^{(t+1)} \leftarrow y_\ell^{(t)} + \eta_y / L \cdot \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)}$

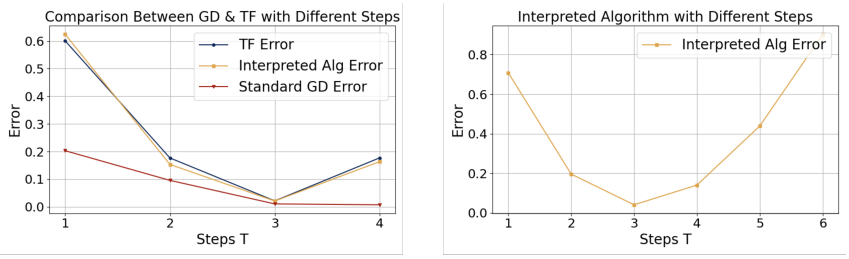
4: **end for**

5: return: $y_q^{(T)}$

We compare a three-layer, two-head transformer with its interpreted algorithm, a three-step GD estimator, and the ridge regression estimator. Figure 4(a) shows that the transformer and its interpreted algorithm produce nearly identical loss curves, with the comparable performance to 3-step GD and ridge estimator (with an around 0.007 gap). The step sizes of the interpreted algorithm are optimized for the given number of steps T. Similarly, the learning rate of the standard GD estimator is optimally searched corresponding to the number of the steps T. Figure 4(b) evaluates the alignment between the transformer and the interpreted algorithm via two measures (Von Oswald et al., 2023): (1) the cosine similarity between their input sensitivity vectors $\partial \hat{y}_{\text{TF}}(x_q)/\partial x_q$ and $\partial \hat{y}_{\text{Alg}}(x_q)/\partial x_q$ (denoted "Model Sim" in the figure); and (2) the L2 norm of their difference, $\|\partial \hat{y}_{\text{TF}}(x_q)/\partial x_q - \partial \hat{y}_{\text{Alg}}(x_q)/\partial x_q\|_2$ (denoted "Grad Diff"). Both metrics confirm a strong agreement: the two- and three-layer models exhibit high cosine similarity and small L2 deviation.

These results demonstrate that the transformer effectively implements the analytically derived algorithm, with behavior nearly indistinguishable from our interpretation.

Divergence with Increasing Steps. We observed a divergent property in the multi-layer looped transformer (**Observation 3**), and we now demonstrate the same behavior in its interpreted algorithm. We use the algorithm interpreted from a 3-layer transformer in the experiments. As shown in Figure 5, the iterative algorithm exhibits divergent after being unrolled 3 times. This stands in sharp contrast to the standard multi-step GD estimator, which is guaranteed to converge for an appropriately chosen step size. We claim that the key insight is that the learned step-sizes are optimized for exactly T steps



(a) (b)

Figure 5: (a) performance comparison on each step (layer for transformer) among different models; (b) performance of 3-layer looped transformer's interpreted algorithm with varying numbers of steps.

(T layers for transformer). When the looped transformer or the algorithm is unrolled beyond T (e.g., $2T$ steps), these step-sizes become too large and the algorithm overshoots the global minimum of the least-squares problem.

Further comparisons with prior work (Von Oswald et al., 2023; He et al., 2025; Ahn et al., 2023), and discussions, including looped vs. original transformers, softmax vs. linear attention, single-layer vs. multi-layer architectures, and varying-covariance vs. constant-covariance regimes, are presented in Appendix C.

5 Application to IV Regression

In this section, we extend our findings on the capabilities of multi-layer transformers to more complex in-context learning tasks, and demonstrate that the multi-layer transformer learns the interpreted algorithm in this scenario.

5.1 Data Generation and Training Settings

Our data generation process mainly follows the prior work (Liang et al., 2024). Suppose we wish to estimate the relationship between a scalar response variable $y_\ell \in \mathbb{R}$ and a p-dimensional predictor $x_\ell \in \mathbb{R}^p$. Unlike the ordinary least squares (OLS) setting, the covariate x_ℓ here is endogenous, *i.e.*, it is correlated with the regression noise ϵ_ℓ. Concretely, we observe $y_\ell = \beta^\top x_\ell + \epsilon_\ell$. This correlation introduces bias when using OLS to estimate β. To address this endogeneity, we introduce an instrumental variable z_ℓ that is correlated with the endogenous variable x_ℓ but uncorrelated with the error term ϵ_ℓ. Under these assumptions, the instrument isolates the exogenous variation in x_ℓ, allowing for consistent inference on β. Specifically, we generate $z_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_q)$, $u_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_u)$, and $\omega_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_\omega)$, and then set

$$x_\ell = \theta^\top z_\ell + \Phi^\top u_\ell + \omega_\ell, \quad y_\ell = \beta^\top x_\ell + \phi^\top u_\ell + \epsilon_\ell, \tag{8}$$

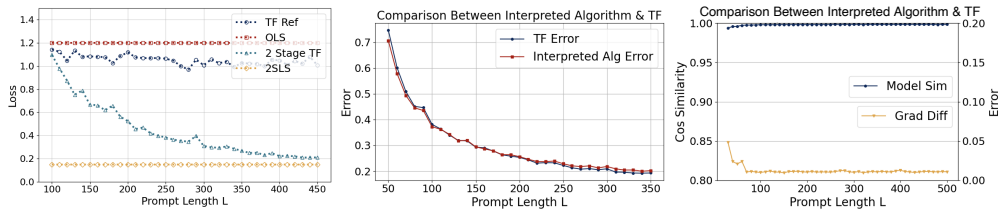
so that u_ℓ serves as the common source of endogeneity between x_ℓ and y_ℓ.

Given the observations $\{(z_\ell, x_\ell, y_\ell)\}_{\ell \in [L]}$, the classical approach to IV regression proceeds in two stages. In the first stage, x_ℓ is regressed on z_ℓ to obtain predicted values $\hat{x}_\ell$. In the second stage, y_ℓ is regressed on $\hat{x}_\ell$. Since $\hat{x}_\ell$ in the second stage depends on the sequence-specific estimator $\hat{\theta}$, its distribution varies across sequences. This aligns with the regime introduced in Section 3, highlighting the practical relevance of our varying covariate distribution setup.

Sequence Generation. To guide the transformer to implement the two-stage estimator, we decompose the procedure into two successive transformer blocks. We define the stage-1 input as

$$Z_{\texttt{ebd}}^{\texttt{st1}} = \begin{bmatrix} z_1 & z_2 & \cdots & z_\ell & z_q \\ x_1 & x_2 & \cdots & x_\ell & 0 \end{bmatrix}, \tag{9}$$

and denote the first- and second-stage blocks by $\texttt{TF}_\theta^{\texttt{st1}}$ and $\texttt{TF}_\theta^{\texttt{st2}}$, respectively. The input sequence for the second stage, $Z_{\texttt{ebd}}^{\texttt{st2}}$, is formed by concatenating the full output sequence of $\texttt{TF}_\theta^{\texttt{st1}}$ (with a stop-gradient applied) with $\{y_\ell\}_{\ell \in [L] \cup \{q\}}$. More details are provided in Appendix B.2.



(a) (b) (c)

Figure 6: Results for IV regression tasks: (a) performance comparison among different models; (b) performance comparison with interpreted algorithm in the second stage; (c) alignment between the transformer and the interpreted algorithm.

Training Settings. $\texttt{TF}_\theta^{\text{st1}}$ is a single-layer and $\texttt{TF}_\theta^{\text{st2}}$ is a three-layer, looped transformer. These two transformers are trained jointly. We define the jointly training loss as a two-part objective:

$$\mathcal{L}(\theta) = \mathbb{E}[(x_q - \texttt{TF}_\theta^{\text{st1}}(Z_{\texttt{ebd}}^{\text{st1}}))^2] + \mathbb{E}[(y_q - \texttt{TF}_\theta^{\text{st2}}(Z_{\texttt{ebd}}^{\text{st2}}))^2]. \tag{10}$$

5.2 Empirical Results of IV Regression

Under our setup, the transformer can effectively solve IV regression tasks. In the first stage, a single-layer transformer learns a GD estimator (He et al., 2025; Von Oswald et al., 2023) and outputs the fitted regressors $\{\hat{x}_\ell\}_{\ell\in[L]\cup\{q\}}$ from the instruments $\{z_\ell\}_{\ell\in[L]\cup\{q\}}$ and endogenous variables $\{x_\ell\}_{\ell\in[L]\cup\{q\}}$. In the second stage, we have the following observation.

Observation 5. The overall performance of the two-block transformer is close to that of the two-stage least squares estimator, and is better than both the OLS estimator and same-depth transformers trained in a single stage. The KQ and OV circuit patterns (Figure 12) and learned algorithm in $\texttt{TF}_\theta^{\text{st2}}$ match the same interpretation in Section 4.

Figure 6(a) shows that the transformer's performance converges to that of the two-stage least squares estimator as the sample size L increases. To confirm that this improvement is due to the two-stage training procedure and not merely to increased model depth, we compare our approach with a single-stage baseline (TF Ref) in which a transformer of the same depth directly regresses y_ℓ on x_ℓ given access to z_ℓ, x_ℓ, and y_ℓ. As shown in Figure 6(a), the transformer trained with our two-stage procedure can reach the comparable performance to the 2SLS, which is the canonical approach for solving IV regression. With the diagonal structured KQ and OV patterns of $\texttt{TF}_\theta^{\text{st2}}$ in Figure 12, we further assess the alignment between the $\texttt{TF}_\theta^{\text{st2}}$ and the interpreted algorithm, as well as their performance, in Figure 6 (b) and (c). The results evaluate that $\texttt{TF}_\theta^{\text{st2}}$ learns the same algorithm as the interpreted algorithm.

Consequently, the multi-layer transformer successfully learns the interpreted algorithm in the second stage, accurately predicting y_q from the fitted regressors $\{\hat{x}_\ell\}_{\ell\in[L]\cup\{q\}}$ and $\{y_\ell\}_{\ell\in[L]}$. This shows that our interpreted algorithm generalizes to other tasks when facing the same situation in our setting. In addition, the second stage aligns our studied regime well, highlighting the practical relevance of our regime.

6 Conclusion

We study in-context linear regression under varying input covariances. Analyzing the model's internal computations reveals a new, depth-dependent algorithm that the transformer learns in this regime. Extending to in-context IV regression, we find that a transformer prompted with a single chain-of-thought step matches two-stage least squares and learns the same algorithm when facing same situations. These results highlight the algorithmic generalization capacity of multi-layer transformers.

Limitations and future work. Our interpretation relies on a population-limit idealization in which the sequence length tends to infinity, allowing the empirical softmax average to be approximated by its expectation under the data distribution. In addition, why the transformer converges to these particular weights under the new regime remains unclear, and we leave this for future work.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research methodology and experiments, and assisting with translation. We have not used generative AI tools for the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for editing software code, and editing a research paper to improve readability. These contents have been checked by two authors. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. 2023.

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

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36:57125–57211, 2023.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and llms by learning to learn discrete functions. *arXiv preprint arXiv:2310.03016*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Siyu Chen, Heejune Sheen, Tianhao Wang, and Zhuoran Yang. Training dynamics of multi-head softmax attention for in-context learning: Emergence, convergence, and optimality. *arXiv preprint arXiv:2402.19442*, 2024.

Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context. *arXiv preprint arXiv:2312.06528*, 2023.

Yingqian Cui, Jie Ren, Pengfei He, Jiliang Tang, and Yue Xing. Superiority of multi-head attention in in-context linear regression. *arXiv preprint arXiv:2401.17426*, 2024.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*, 2022.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

George Fikioris. Spectral properties of kac–murdock–szegö matrices with a complex parameter. *Linear Algebra and its Applications*, 553:182–210, 2018.

Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. *Advances in Neural Information Processing Systems*, 37:98675–98716, 2024.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2026.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023.

Angeliki Giannou, Liu Yang, Tianhao Wang, Dimitris Papailiopoulos, and Jason D Lee. How well can transformers emulate in-context newton's method? *arXiv preprint arXiv:2403.03183*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Jianliang He, Xintian Pan, Siyu Chen, and Zhuoran Yang. In-context linear regression demystified: Training dynamics and mechanistic interpretability of multi-head softmax attention. *arXiv preprint arXiv:2503.12734*, 2025.

Xinyang Hu, Fengzhuo Zhang, Siyu Chen, and Zhuoran Yang. Unveiling the statistical foundations of chain-of-thought prompting methods. *arXiv preprint arXiv:2408.14511*, 2024.

Hong Jun Jeon, Jason D Lee, Qi Lei, and Benjamin Van Roy. An information-theoretic analysis of in-context learning. *arXiv preprint arXiv:2401.15530*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Hongkang Li, Songtao Lu, Xiaodong Cui, Pin-Yu Chen, and Meng Wang. Understanding mamba in in-context learning with outliers: A theoretical generalization analysis. In *High-dimensional Learning Dynamics 2025*.

Yingcong Li, Davoud Ataee Tarzanagh, Ankit Singh Rawat, Maryam Fazel, and Samet Oymak. Gating is weighting: Understanding gated linear attention through in-context learning. *arXiv preprint arXiv:2504.04308*, 2025.

Haodong Liang, Krishnakumar Balasubramanian, and Lifeng Lai. Transformers handle endogeneity in in-context linear regression. *arXiv preprint arXiv:2410.01265*, 2024.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*, 2023.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

Eshaan Nichani, Alex Damian, and Jason D Lee. How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*, 2024.

Feng Nie, Meixi Chen, Zhirui Zhang, and Xu Cheng. Improving few-shot performance of language models via nearest neighbor calibration. *arXiv preprint arXiv:2212.02216*, 2022.

Junsoo Oh, Wei Huang, and Taiji Suzuki. Mamba can learn low-dimensional targets in-context via test-time feature learning. *arXiv preprint arXiv:2510.12026*, 2025.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Core Francisco Park, Andrew Lee, Ekdeep Singh Lubana, Yongyi Yang, Maya Okawa, Kento Nishi, Martin Wattenberg, and Hidenori Tanaka. Iclr: In-context learning of representations. *arXiv preprint arXiv:2501.00070*, 2024.

Heejune Sheen, Siyu Chen, Tianhao Wang, and Harrison H Zhou. Implicit regularization of gradient flow on one-layer softmax attention. *arXiv preprint arXiv:2403.08699*, 2024.

Aaditya Singh, Stephanie Chan, Ted Moskovitz, Erin Grant, Andrew Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 36, 2023.

Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. Privacy-preserving in-context learning with differentially private few-shot generation. *arXiv preprint arXiv:2309.11765*, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers learn temporal difference methods for in-context reinforcement learning. *arXiv preprint arXiv:2405.13861*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024. *URL https://arxiv. org/abs/2312.06635*, pp. 1.

Ruiqi Zhang, Jingfeng Wu, and Peter Bartlett. In-context learning of a linear transformer block: benefits of the mlp component and one-step gd initialization. *Advances in Neural Information Processing Systems*, 37:18310–18361, 2024.

Yedi Zhang, Aaditya K Singh, Peter E Latham, and Andrew Saxe. Training dynamics of in-context learning in linear attention. *arXiv preprint arXiv:2501.16265*, 2025.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*, 2023.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

A Derivations of Methodology in Section 4

A.1 Mechanistic Interpretation via Gaussian Population

Setup and population-limit definition. Consider a transformer whose input consists of L in-context tokens $\{z_\ell^{(0)}\}_{\ell\in[L]} \overset{\text{i.i.d.}}{\sim} p_0 := \mathcal{N}(\mu, \Sigma)$ with $\Sigma \succ 0$, together with a query token $z_q^{(0)} \in \mathbb{R}^{d+1}$ that is fixed and carries no distributional assumption (in our setup $z_q^{(0)} = (x_q, 0)$ has a degenerate last entry). Under the prefix mask of Section 3, every token $\ell \in [L] \cup \{q\}$ attends to the same L in-context keys.

We analyze the attention layer in the *population limit*, the $L \to \infty$ idealization in which the empirical softmax average over the in-context keys is replaced by its expectation under the layer-t in-context distribution $p_t := \mathcal{N}(\mu^{(t)}, \Sigma^{(t)})$ with $\Sigma^{(t)} \succ 0$. Concretely, for any token index $\ell \in [L] \cup \{q\}$ and head $h \in [H]$,

$$\begin{aligned}\Psi^{(t+1,h)} Z_{1:L}^{(t)} \cdot \texttt{smax}\big(Z_{1:L}^{(t)\top} \Phi^{(t+1,h)} z_\ell^{(t)}\big) &= \frac{\frac{1}{L}\sum_{\iota=1}^{L} \Psi^{(t+1,h)} z_\iota^{(t)} \exp\big(z_\iota^{(t)\top} \Phi^{(t+1,h)} z_\ell^{(t)}\big)}{\frac{1}{L}\sum_{\iota'=1}^{L} \exp\big(z_{\iota'}^{(t)\top} \Phi^{(t+1,h)} z_\ell^{(t)}\big)} \\ &\overset{L\to\infty}{=} \frac{\mathbb{E}_{z'\sim p_t}\big[\Psi^{(t+1,h)} z' \exp(z'^\top \Phi^{(t+1,h)} z_\ell^{(t)})\big]}{\mathbb{E}_{z''\sim p_t}\big[\exp(z''^\top \Phi^{(t+1,h)} z_\ell^{(t)})\big]},\end{aligned} \tag{11}$$

where the limit holds by the law of large numbers at rate $O_p(L^{-1/2})$. Two structural features of equation 11 drive the analysis below: the averages run over the in-context indices $\iota \in [L]$ alone, so the query's distribution never enters; and the query position $\ell = q$ enters equation 11 only as the fixed vector $\Phi^{(t+1,h)} z_q^{(t)}$ inside the exponent, never as a measure to be integrated.

Proof of Theorem 1. We first recall the notation used throughout. The pair (μ, Σ) is the initial mean and covariance of the in-context distribution $p_0 = \mathcal{N}(\mu, \Sigma)$. The layer-$t$ Gaussian-moment pair $(\mu^{(t)}, \Sigma^{(t)})$ is defined recursively by $(\mu^{(0)}, \Sigma^{(0)}) = (\mu, \Sigma)$ and the propagation rule of Theorem 1, namely $\mu^{(t+1)} = A^{(t)}\mu^{(t)} + b^{(t)}$ and $\Sigma^{(t+1)} = A^{(t)}\Sigma^{(t)}(A^{(t)})^\top$, where the layer-$t$ map $(A^{(t)}, b^{(t)})$ is itself a function of $(\mu^{(t)}, \Sigma^{(t)})$ and the trained circuits. The matrices $\Phi^{(t+1,h)}, \Psi^{(t+1,h)} \in \mathbb{R}^{(d+1)\times(d+1)}$ are the layer-$(t+1)$, head-h KQ and OV circuits as defined in Section 3; they are the only trainable parameters and are fixed throughout the analysis. We write $p_t := \mathcal{N}(\mu^{(t)}, \Sigma^{(t)})$ for the layer-t in-context distribution. Empirical softmax averages over the in-context keys are interpreted in the population limit defined by equation 11.

We argue by induction on the layer index $t \in \{0, 1, \ldots, T-1\}$. At each layer we maintain the following two-part inductive hypothesis (IH):

(IH-a) *In-context tokens.* $\{z_\ell^{(t)}\}_{\ell\in[L]} \overset{\text{i.i.d.}}{\sim} p_t = \mathcal{N}(\mu^{(t)}, \Sigma^{(t)})$, and $\Sigma^{(t)} \succ 0$.

(IH-b) *Query token.* $z_q^{(t)} \in \mathbb{R}^{d+1}$ is a fixed vector; no distributional claim is made about it.

The induction shows that the layer-t population-limit update produces, identically for every token index $\ell \in [L] \cup \{q\}$, the affine recursion $z_\ell^{(t+1)} = A^{(t)} z_\ell^{(t)} + b^{(t)}$, and that (IH-a) and (IH-b) are preserved at level $t+1$.

Base case ($t = 0$). Assumption (*i*) of Theorem 1 gives (IH-a) directly with $\mu^{(0)} = \mu$ and $\Sigma^{(0)} = \Sigma \succ 0$. Assumption (*ii*) gives (IH-b) with $z_q^{(0)}$ as specified.

Inductive step. Suppose (IH-a) and (IH-b) hold at layer t. Fix a token index $\ell \in [L] \cup \{q\}$ and a head $h \in [H]$. We compute the head-h contribution to the position-ℓ output of layer $t+1$ in four steps.

Step 1 (replacing the empirical average by a population expectation). By the population-limit identity equation 11, the head-h contribution to the (residual-stripped) position-ℓ output is the

empirical softmax average

$$\frac{\frac{1}{L}\sum_{\iota=1}^{L}\Psi^{(t+1,h)}z_{\iota}^{(t)}\exp\big(z_{\iota}^{(t)\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)}{\frac{1}{L}\sum_{\iota'=1}^{L}\exp\big(z_{\iota'}^{(t)\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)}.$$

By (IH-a), the in-context sample $\{z_{\iota}^{(t)}\}_{\iota\in[L]}$ is i.i.d. from p_t. Numerator and denominator are each empirical means of the form $\frac{1}{L}\sum_{\iota=1}^{L}f(z_{\iota}^{(t)})$ with i.i.d. summands and finite second moment (the integrand is the product of an exponential of a quadratic-plus-linear form in a Gaussian random vector, which has all moments finite under $\Sigma^{(t)}\succ 0$). The law of large numbers therefore yields

$$\Psi^{(t+1,h)}Z_{1:L}^{(t)}\cdot\mathtt{smax}\big(Z_{1:L}^{(t)\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\ \overset{L\to\infty}{=}\ \frac{\mathbb{E}_{z'\sim p_t}\big[\Psi^{(t+1,h)}z'\,\exp\big(z'^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\big]}{\mathbb{E}_{z''\sim p_t}\big[\exp\big(z''^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\big]}. \quad (12)$$

The position-ℓ value $z_{\ell}^{(t)}$ enters the right-hand side of equation 12 only as the fixed vector $\Phi^{(t+1,h)}z_{\ell}^{(t)}$ inside the exponent, i.e. as a parameter rather than as a variable of integration. Consequently equation 12 holds identically whether ℓ refers to an in-context position or to the query, and (IH-b) is not invoked.

Step 2 (exponential tilting). Writing $p_t(z')\propto\exp\big(-\frac{1}{2}\|z'-\mu^{(t)}\|^2_{(\Sigma^{(t)})^{-1}}\big)$ with $\|v\|_M^2:=v^{\top}Mv$, the integrands of numerator and denominator in equation 12 share the exponential weight

$$\exp\big(z'^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\,p_t(z')\ \propto\ \exp\Big(z'^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}-\tfrac{1}{2}\|z'-\mu^{(t)}\|^2_{(\Sigma^{(t)})^{-1}}\Big).$$

Completing the square in z', with $\Phi^{(t+1,h)}z_{\ell}^{(t)}$ held fixed, gives the identity

$$z'^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}-\tfrac{1}{2}\|z'-\mu^{(t)}\|^2_{(\Sigma^{(t)})^{-1}}\ =\ -\tfrac{1}{2}\big\|z'-\tilde{\mu}_{\ell}^{(t,h)}\big\|^2_{(\Sigma^{(t)})^{-1}}+C_{\ell}^{(t,h)},$$

where the shifted mean and the constant (z'-independent) term are

$$\tilde{\mu}_{\ell}^{(t,h)}\ :=\ \mu^{(t)}+\Sigma^{(t)}\Phi^{(t+1,h)}z_{\ell}^{(t)},\qquad C_{\ell}^{(t,h)}\ :=\ \tfrac{1}{2}\big\|\tilde{\mu}_{\ell}^{(t,h)}\big\|^2_{(\Sigma^{(t)})^{-1}}-\tfrac{1}{2}\|\mu^{(t)}\|^2_{(\Sigma^{(t)})^{-1}}.$$

Since $\Sigma^{(t)}\succ 0$ by (IH-a), the quadratic in z' on the right-hand side is strictly negative-definite, so both integrals in equation 12 converge absolutely and the constant $C_{\ell}^{(t,h)}$ cancels between numerator and denominator. The tilted measure

$$\tilde{p}_t^{(h)}(z')\ :=\ \frac{p_t(z')\exp\big(z'^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)}{\mathbb{E}_{z''\sim p_t}\big[\exp\big(z''^{\top}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\big]}\ =\ \mathcal{N}\big(\tilde{\mu}_{\ell}^{(t,h)},\,\Sigma^{(t)}\big) \quad (13)$$

is therefore a well-defined Gaussian density on $\mathbb{R}^{d+1}$, and the right-hand side of equation 12 simplifies, by linearity of expectation, to

$$\mathbb{E}_{z'\sim\tilde{p}_t^{(h)}}\big[\Psi^{(t+1,h)}z'\big]\ =\ \Psi^{(t+1,h)}\tilde{\mu}_{\ell}^{(t,h)}\ =\ \Psi^{(t+1,h)}\big(\mu^{(t)}+\Sigma^{(t)}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big).$$

Step 3 (assembling the layer-t output). Summing the head contributions and adding the residual connection $z_{\ell}^{(t+1)}=z_{\ell}^{(t)}+\mathtt{Attn}_{\theta}(\cdot)_{\ell}$ from Section 3,

$$\begin{aligned}z_{\ell}^{(t+1)}&=z_{\ell}^{(t)}+\sum_{h=1}^{H}\Psi^{(t+1,h)}\big(\mu^{(t)}+\Sigma^{(t)}\Phi^{(t+1,h)}z_{\ell}^{(t)}\big)\\&=\Big(I_{d+1}+\sum_{h=1}^{H}\Psi^{(t+1,h)}\Sigma^{(t)}\Phi^{(t+1,h)}\Big)z_{\ell}^{(t)}+\sum_{h=1}^{H}\Psi^{(t+1,h)}\mu^{(t)}\\&=:\ A^{(t)}z_{\ell}^{(t)}+b^{(t)},\end{aligned} \quad (14)$$

which is the affine recursion claimed in Theorem 1. Steps 1–3 fixed an arbitrary token index $\ell\in[L]\cup\{q\}$ but never used (IH-b); the same affine map $(A^{(t)},b^{(t)})$ therefore applies uniformly to every position, including the query.

Step 4 (preserving the inductive hypothesis at layer $t+1$). We verify (IH-a) and (IH-b) at level $t+1$.

(*IH-a at level $t+1$.*) The in-context tokens at layer $t+1$ are the images $\{A^{(t)}z_\ell^{(t)} + b^{(t)}\}_{\ell\in[L]}$ of the i.i.d. Gaussian sample $\{z_\ell^{(t)}\}_{\ell\in[L]} \overset{\text{i.i.d.}}{\sim} p_t$ under the deterministic affine map $z \mapsto A^{(t)}z + b^{(t)}$. Each $z_\ell^{(t+1)}$ is a measurable function of the corresponding $z_\ell^{(t)}$ alone, so mutual independence of the inputs is inherited by the outputs; identical distribution is preserved because the same map is applied to every ℓ. The affine image of a Gaussian random vector is Gaussian, with mean and covariance

$$\mu^{(t+1)} \;=\; A^{(t)}\mu^{(t)} + b^{(t)}, \qquad \Sigma^{(t+1)} \;=\; A^{(t)}\Sigma^{(t)}\big(A^{(t)}\big)^\top. \tag{15}$$

By (IH-a), $\Sigma^{(t)} \succ 0$, and by Assumption (*iii*) of Theorem 1, $A^{(t)}$ is non-singular. The conjugation $M \mapsto AMA^\top$ of a positive-definite matrix by an invertible matrix is positive-definite, so $\Sigma^{(t+1)} \succ 0$. Hence $\{z_\ell^{(t+1)}\}_{\ell\in[L]} \overset{\text{i.i.d.}}{\sim} p_{t+1} := \mathcal{N}(\mu^{(t+1)}, \Sigma^{(t+1)})$, restoring (IH-a).

(*IH-b at level $t+1$.*) The query at level $t+1$ is $z_q^{(t+1)} = A^{(t)}z_q^{(t)} + b^{(t)} \in \mathbb{R}^{d+1}$, a fixed vector by (IH-b) at level t; (IH-b) at level $t+1$ holds vacuously.

This completes the inductive step and hence the proof. □

To summarize, in the population limit the multi-layer attention transformer is exactly described by the recursion

$$\begin{aligned} &z_\ell^{(t+1)} = A^{(t)}z_\ell^{(t)} + b^{(t)}, \quad \forall(\ell,t) \in \big([L]\cup\{q\}\big)\times[T-1], \\ &\quad\text{where}\quad A^{(t)} = I_{d+1} + \sum_{h=1}^{H}\Psi^{(t+1,h)}\Sigma^{(t)}\Phi^{(t+1,h)}, \quad b^{(t)} = \sum_{h=1}^{H}\Psi^{(t+1,h)}\mu^{(t)}, \\ &\qquad \mu^{(t+1)} = A^{(t)}\mu^{(t)} + b^{(t)}, \;\; \Sigma^{(t+1)} = A^{(t)}\Sigma^{(t)}{A^{(t)}}^\top, \;\; \mu^{(0)} = \mu, \;\; \Sigma^{(0)} = \Sigma. \end{aligned} \tag{16}$$

Finite-L approximation. The actual finite-L softmax attention differs from equation 16 by an additive residual $r_\ell^{(t)}(L)$ arising from the empirical softmax average:

$$\begin{aligned} \Psi^{(t+1,h)}Z_{1:L}^{(t)}\cdot \texttt{smax}\big(Z_{1:L}^{(t)\top}\Phi^{(t+1,h)}z_\ell^{(t)}\big) &= \frac{\frac{1}{L}\sum_{\iota=1}^{L}\Psi^{(t+1,h)}z_\iota^{(t)}\cdot\exp\big(z_\iota^{(t)\top}\Phi^{(t+1,h)}z_\ell^{(t)}\big)}{\frac{1}{L}\sum_{\iota'=1}^{L}\exp\big(z_{\iota'}^{(t)\top}\Phi^{(t+1,h)}z_\ell^{(t)}\big)} \\ &= \frac{\mathbb{E}_{z'\sim p_t}\big[\Psi^{(t+1,h)}z'\cdot\exp(z'^\top\Phi^{(t+1,h)}z_\ell^{(t)})\big]}{\mathbb{E}_{z''\sim p_t}\big[\exp(z''^\top\Phi^{(t+1,h)}z_\ell^{(t)})\big]} + r_\ell^{(t)}(L), \end{aligned}$$

with $\|r_\ell^{(t)}(L)\| = O_p(L^{-1/2})$ by sub-Gaussian concentration of the empirical softmax around its expectation. We take this as a modeling approximation. Figure 3 (b) confirms that, when L increases, the empirical transformer output and the population-limit prediction equation 16 are nearly identical.

A.2 Simplification under Emerged Patterns

Recall that the learned looped transformer satisfies the following patterns:

$$\Phi^{(h)} = \begin{bmatrix}\omega^{(h)}I_d & 0_d\\ 0_d^\top & 0\end{bmatrix}, \quad \Psi^{(h)} = \begin{bmatrix}u_1^{(h)}I_d & 0_d\\ 0_d^\top & u_2^{(h)}\end{bmatrix}, \quad \forall(h)\in[H]. \tag{17}$$

Denote $\omega = (\omega^{(1)},\ldots,\omega^{(H)}) \in \mathbb{R}^H$, $u_1 = (u_1^{(1)},\ldots,u_1^{(H)}) \in \mathbb{R}^H$, and $u_2 = (u_2^{(1)},\ldots,u_2^{(H)}) \in \mathbb{R}^H$. Moreover, under the embedding and data generation process in Section 3, we have

$$z_\ell = [x_\ell \;\; y_\ell]^\top \in \mathbb{R}^{d+1}, \;\; \text{where} \;\; x_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0_d,\Sigma), \;\; y_\ell = \langle x_\ell,\beta\rangle + \epsilon_\ell \;\; \text{with} \;\; \epsilon \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,\sigma^2).$$

Let Γ denote the covariance matrix of z_ℓ. Hence, by simple calculation, we have $\mu = \mathbb{E}[z_\ell] = 0$ and

$$\Gamma = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{12}^\top & \Gamma_{22} \end{bmatrix} \in \mathbb{R}^{(d+1)\times(d+1)},$$
$$\text{where } \Gamma_{11} = \Sigma \in \mathbb{R}^{d\times d}, \quad \Gamma_{12} = \Sigma\beta \in \mathbb{R}^d, \quad \Gamma_{22} = \|\beta\|_\Sigma^2 + \sigma^2 \in \mathbb{R}.$$

Next, we simplify the recuisive form in equation 16 based on the specific parametrization above. Note that $\mu_0 = 0_{d+1}$. Suppose that $\mu^{(t)} = 0_{d+1}$, by a simple induction argument, it holds that

$$b^{(t)} = \sum_{h=1}^{H} \Phi^{(h)}\mu^{(t)} = 0_{d+1} \text{ s.t. } \mu^{(t+1)} = A^{(t)}\mu^{(t)} + b^{(t)} = 0_{d+1},$$

which implies that $\mu^{(t)} = 0_{d+1}$ for all $t \in [T]$. Then, we can reduce equation 16 to

$$z_\ell^{(t+1)} = A^{(t)} z_\ell^{(t)}, \quad \forall (\ell, t) \in [L] \cup \{q\} \times [T-1],$$
$$\text{where} \quad A^{(t)} = I_d + \sum_{h=1}^{H} \Psi^{(h)}\Gamma^{(t)}\Phi^{(h)}, \quad \Gamma^{(t+1)} = A^{(t)}\Gamma^{(t)}{A^{(t)}}^\top \text{ with } \Gamma^{(0)} = \Gamma. \tag{18}$$

Next, we derive the recursive form of covariance. First, based on equation 17 and equation 18, note that

$$A^{(t)} = I_{d+1} + \sum_{h=1}^{H} \Psi^{(h)}\Gamma^{(t)}\Phi^{(h)} := \begin{bmatrix} A_{11}^{(t)} & 0_d \\ A_{21}^{(t)} & 1 \end{bmatrix}$$
$$\text{where } A_{11}^{(t)} = I_d + \sum_{h=1}^{H} u_1^{(h)}\omega_1^{(h)} \cdot \Gamma_{11}^{(t)} \text{ and } A_{21}^{(t)} = \sum_{h=1}^{H} u_2^{(h)}\omega_1^{(h)} \cdot {\Gamma_{12}^{(t)}}^\top.$$

Using this block decomposition in the layer-wise iteration rule equation 16, we obtain in the population limit

$$\begin{aligned} z_\ell^{(t+1)} = \begin{bmatrix} x_\ell^{(t+1)} \\ y_\ell^{(t+1)} \end{bmatrix} = A^{(t)} \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} &= \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \begin{bmatrix} \langle u_1, \omega\rangle \cdot \Gamma_{11}^{(t)} \cdot x_\ell^{(t)} \\ \langle u_2, \omega\rangle \cdot {\Gamma_{12}^{(t)}}^\top \cdot x_\ell^{(t)} \end{bmatrix} \\ &\overset{L\to\infty}{=} \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \begin{bmatrix} \frac{\langle u_1,\omega\rangle}{L} \sum_{\ell'=1}^{L} x_{\ell'}^{(t)} {x_{\ell'}^{(t)}}^\top \cdot x_\ell^{(t)} \\ \frac{\langle u_2,\omega\rangle}{L} \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} {x_{\ell'}^{(t)}}^\top \cdot x_\ell^{(t)} \end{bmatrix}, \end{aligned} \tag{19}$$

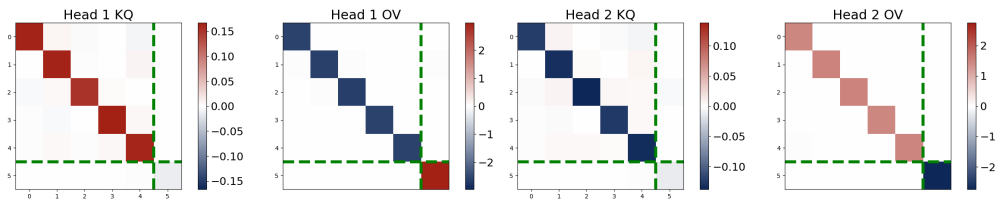
where the last identity replaces the population sub-blocks $\Gamma_{11}^{(t)} = \mathbb{E}[xx^\top]$ and $\Gamma_{12}^{(t)} = \mathbb{E}[xy]$ by their empirical sample averages over the L in-context tokens, which agree with the population values up to an $O_p(L^{-1/2})$ residual; the sample-average form is the one actually computed by the finite-L transformer and matches Algorithm 1.

B More Training Details and Settings

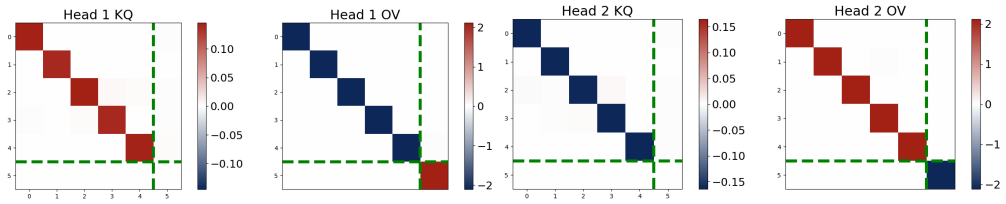
B.1 Extensive Experiments on the Linear Regression Task

In this subsection, we conduct experiments with a single-head transformer, increased dimensionality, and a broader range of covariance eigenvalues to evaluate the robustness and consistency of our conclusions. We also provide several visualizations that were omitted from the main paper due to space constraints. The corresponding results are presented below.

Additional KQ and OV Visualizations. We extend our analysis beyond the three-layer transformer presented in the main paper by providing KQ and OV visualizations for the two- and four-layer models in Figure 7. These visualizations show that the diagonal patterns learned by the model are highly consistent across depths, indicating a generalizable learning mechanism. This finding reinforces our main conclusion that transformers learn a specific algorithmic structure regardless of the number of layers.



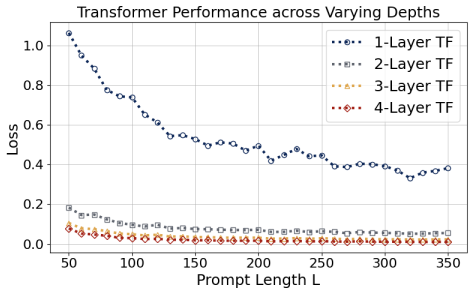
(a) 2-Layer Looped Transformer



(b) 4-Layer Looped Transformer

Figure 7: The learned patterns of 2- and 4-layer looped Transformer model.

Performance Comparison on an Absolute Scale. We plot the absolute performance differences among the 1-, 2-, 3-, and 4-layer transformers in Figure 8. The figure clearly shows a substantial gap between the 1-layer transformer and the multi-layer models. Moreover, when the number of layers exceeds three, the additional performance improvement becomes negligible (less than 0.01).



(a)

Figure 8: The absolute performance comparison among 1-, 2-, 3-, and 4-layer transformer.

Performance of Single-head Transformers. We also conduct experiments on transformers with a single attention head. As shown in Figure 9, we use a two-layer model as a representative example; all other settings are identical to those in the main paper. We observe that the learned KQ and OV circuit patterns are structurally similar to those in multi-head models, but the learned inner products $\langle u_1, \omega \rangle$ and $\langle u_2, \omega \rangle$ differ. This difference in the inner products leads to inferior performance, as seen in Figure 9(b). This phenomenon is consistent with findings from (He et al., 2025), who noted that single-head attention corresponds to a less expressive, non-parametric predictor. Our results demonstrate that this limitation persists in multi-layer transformers and leads to reduced performance.

Larger Dimension. Figure 10 shows the performance and learned patterns of a 3-layer, 2-head transformer with dimensionality $d = 12$. The KQ and OV circuit patterns remain consistent with those observed in the $d = 5$ case (Figure 10(a)). Moreover, the transformer closely aligns with the interpreted algorithm (Figures 10(b) and (c)), with alignment evaluated as in Section 4. These results demonstrate that our conclusions generalize robustly to higher-dimensional settings.

Broader Range of Covariance Eigenvalues. We further examine the impact of varying the covariance eigenvalues λ_i used in data generation by considering a wider range, specifically $\lambda_{\text{min}} = 0.5$ and $\lambda_{\text{max}} = 3$. A 3-layer, 2-head transformer is used for this experiment. Figure 11(a) visualizes

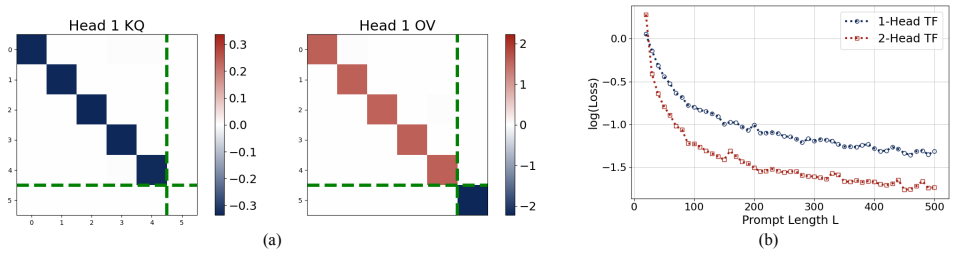


Figure 9: The results of the single head transformer: (a) The patterns of KQ and OV circuits; (b) ICL loss comparison between single-head and multi-head.

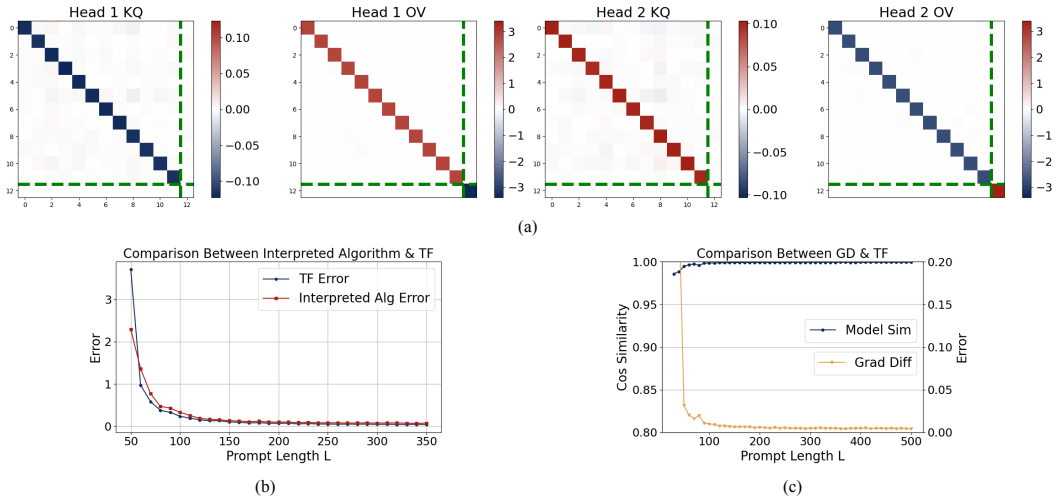


Figure 10: (a) The learned patterns of a 3-layer, 2-head Transformer model when the dimension $d = 12$. (b) Performance comparison between the transformer and the interpreted algorithm. (c) Alignment between the transformer and the interpreted algorithm.

the learned KQ and OV circuits, (b) compares the performance of the trained transformer and the interpreted algorithm, and (c) quantifies their alignment. These results provide additional validation of the proposed interpreted algorithm and yield conclusions consistent with those obtained under the eigenvalue range used in the main paper, demonstrating the robustness of the interpreted algorithm to a broader range of covariance eigenvalues.

B.2 Training Details for the IV Regression Task

Sequence Generation Details. In this section, we describe the process of generating training sequences for the IV regression task. Recall that the input sequences in stage 1 and stage 2 are $\{(z_\ell, x_\ell)\}_{\ell\in[L]\cup\{q\}}$ and $\{(\hat{x}_\ell, y_\ell)\}_{\ell\in[L]\cup\{q\}}$, respectively. The variables x_ℓ and y_ℓ are computed according to Equation equation 8. During training, x_q and y_q are replaced by zero tokens. A key step in the two-stage training procedure is obtaining $\{\hat{x}_\ell\}_{\ell\in[L]\cup\{q\}}$. To this end, we employ a single-layer, two-head softmax-attention transformer in the first stage. The first-stage task is to regress x_ℓ on z_ℓ, which corresponds to a linear regression problem with covariates sampled from a standard Gaussian distribution. For this task, the transformer learns a GD-style predictor once the sequence length L is sufficiently large (He et al., 2025). Specifically, for all $\ell \in [L] \cup \{q\}$, we have $\hat{x}_\ell = \frac{1}{L}\sum_{\ell'=1}^{L} x_{\ell'} z_{\ell'}^\top \cdot z_\ell$. Thus, applying the trained first-stage transformer yields fitted regressors $\hat{x}_\ell$ that are insulated from the noise terms u_ℓ and ω_ℓ. In practice, unlike prior work that only considers the query position as the output, we take the entire sequence as the output. Each input token is prevented from attending to itself at inference time via a prefix mask. However, during

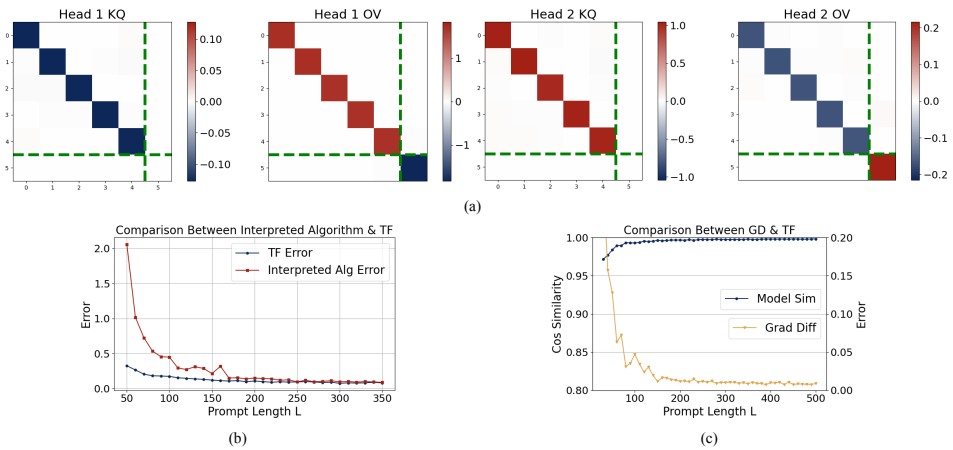


Figure 11: (a) The learned patterns of a 3-layer, 2-head Transformer model when the range of eigenvalues is broader. (b) Performance comparison between the transformer and the interpreted algorithm. (c) Alignment between the transformer and the interpreted algorithm.

Algorithm 2 One Iteration for Training In-Context IV Regression Tasks

Require: Sequence Length L, Task parameters θ, Φ, β, ϕ, Transformer Blocks $\text{TF}_\theta^{\text{st1}}$, $\text{TF}_\theta^{\text{st2}}$.

for $\ell = 1, \ldots, L$ **do**
  Sample: $z_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_q), u_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_u), \omega_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_\omega), \epsilon_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_\epsilon)$
  Compute: $x_\ell = \theta^\top z_\ell + \Phi^\top u_\ell + \omega_\ell$
  Compute: $y_\ell = \beta^\top x_\ell + \phi^\top u_\ell + \epsilon_\ell$ {According to equation 8}
end for
Sample: $z_q \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_q), \omega_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_\omega), \epsilon_\ell \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \Sigma_\epsilon)$
Compute: $x_q = \theta^\top z_q$
Compute: $y_q = \beta^\top x_q$ {Generate query token}
Concatenate: $Z_{\text{ebd}}^{\text{st1}} \leftarrow \{(\{z_\ell\}_{\ell \in [L]}, z_q), (\{x_\ell\}_{\ell \in [L]}, 0)\}$
Inference: $(\{\hat{x}_\ell\}_{\ell \in [L]}, \hat{x}_q) \leftarrow \text{TF}^{\text{st1}}$ {Gradient stop}
Concatenate: $Z_{\text{ebd}}^{\text{st2}} \leftarrow \{(\{\hat{x}_\ell\}_{\ell \in [L]}, \hat{x}_q), (\{y_\ell\}_{\ell \in [L]}, 0)\}$
Calculate: loss = $\mathcal{L}_{mse}(x_q, \text{TF}_\theta^{\text{st1}}(Z_{\text{ebd}}^{\text{st1}})) + \mathcal{L}_{mse}(y_q, \text{TF}_\theta^{\text{st2}}(Z_{\text{ebd}}^{\text{st2}}))$
loss.backward(), optimizer.step()
return: $\text{TF}_\theta^{\text{st1}}$, $\text{TF}_\theta^{\text{st2}}$

stage-1 training, we still compute the loss only on the final token, as defined in Equation equation 10. Pseudo-code for sequence generation and training of IV regression tasks is provided in Algorithm 2.

Training Details. For hyperparameters, we set the instrument dimension of z_ℓ to $q = 6$ and the endogenous variable dimension of x_ℓ to $p = 4$. The covariance matrices Σ_u, Σ_ω, and Σ_ϵ are all the identity matrix. The sequence length is $L = 200$, and the model is trained for 200,000 iterations. The first-stage transformer is single-layer, and the second-stage transformer is a three-layer looped transformer. All other hyperparameters are the same as those in Section 4.

Learned Patterns of KQ and OV Circuit. We visualized the learned KQ and OV patterns of the second-stage transformer in Figure 12. The patterns are similar to those learned in our linear regression experiments with varying covariate distributions, suggesting that the transformer learns the same interpreted algorithm to solve the second stage of the IV regression task. In Section 5, we evaluate the alignment between the transformer's behavior and our interpreted algorithm. Our experimental results show a high degree of alignment, suggesting the multi-layer transformer can be applied to more complex scenarios.

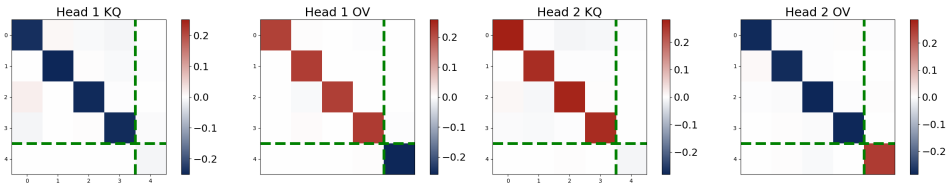


Figure 12: Visualizations of KQ and OV circuit patterns of the transformer in the second stage.

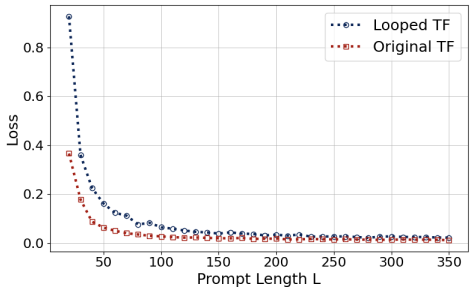


Figure 13: Performance comparison between looped and original transformer.

C Discussions and Extensions

C.1 Looped Transformer vs. Original Transformer

In this subsection, we study the connection between the looped transformer and the transformer without weight tying (the *original* transformer). The experimental settings for the original transformer are identical to those described in Section 4 but without weight tying across layers, and we use a 2-layer transformer as a representative example. We observe two distinct behaviors depending on the random initialization: **(1)** both the KQ and OV patterns are diagonal and resemble Equation equation 5 (the *Structured* case); **(2)** the KQ and OV patterns appear disordered (the *Disordered* case). The corresponding patterns and ICL prediction performance are shown in Figure 21. When the KQ and OV patterns are diagonal, the learned behavior can be readily interpreted as Algorithm 1 with layer-dependent step sizes $\{\eta_x^t\}_{t=1}^T$ and $\{\eta_y^t\}_{t=1}^T$ (obtained by adding the layer superscript t to Φ and Ψ in each layer). Figure 21(d) provides empirical evidence supporting this interpretation. When the KQ and OV patterns are disordered, however, we find that the transformer surprisingly still achieves strong ICL performance (Figure 21(c)). This suggests that multi-layer transformers can successfully handle linear regression tasks even when the learned patterns deviate from a strictly diagonal structure.

We attribute this phenomenon to the increased representational flexibility of multi-layer transformers without weight tying. Such models can implement multiple distinct solution strategies for the linear regression task, with the choice of strategy strongly influenced by the initialization. Certain random seeds lead to diagonal patterns and recover Algorithm 1, whereas others give rise to alternative algorithmic solutions. In contrast, the multi-layer looped transformer, due to its shared weights across layers, has a more constrained hypothesis space and consistently converges to Algorithm 1, regardless of the initialization.

In addition, we compare the validation ICL loss of the looped and original transformers in Figure 13. Their performance converges as the sequence length L increases, indicating that weight tying does not negatively impact transformer performance. The multi-layer looped transformer therefore retains sufficient expressivity to solve the linear regression task in our regime. Moreover, its simplified structure enables it to effectively learn the consistent algorithm (the interpreted algorithm) required for this task.

C.2 Softmax Attention vs. Linear Attention

Previous works (Von Oswald et al., 2023; Ahn et al., 2023) have investigated the mechanisms of linear-attention transformers on linear regression tasks with constant input covariance Σ. This architecture, which removes the nonlinear activation, can be simplified to

$$\mathtt{TF}_\theta(Z_{\mathtt{ebd}}) = Z_{\mathtt{ebd}} + \frac{1}{L}\sum_{h=1}^{H} \Psi^{(h)} Z_{\mathtt{ebd}} \cdot M \cdot Z_{\mathtt{ebd}}^\top \Phi^{(h)} Z_{\mathtt{ebd}}, \quad M = \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}, \tag{20}$$

where M is the prefix mask, playing the same role as $\mathtt{msk}(\cdot)$ in Equation equation 2, and L is the sequence length. Prior studies have shown that a single linear-attention layer can implement one step of GD, while a multi-layer linear transformer can implement a GD++ algorithm when the input covariance is the identity matrix (Von Oswald et al., 2023).

In this study, we extend this analysis to looped multi-layer linear transformers, investigating their behavior in the varying-covariance regime and establishing a connection between linear and softmax transformers. We adapt the linear-attention architecture from (Ahn et al., 2023), with one key modification: we make all attention parameters trainable, whereas the original work fixes the final entries of the KQ and OV circuits. This ensures that the linear and softmax architectures share the same structure and training objective.

The input data construction and training settings follow the methodology in Sections 3 and 4, respectively. Using a 2-layer, 1-head linear transformer as a representative example, we observe that the model develops structural patterns similar to those observed with softmax transformers in the main paper, as shown in Figure 14(a). Furthermore, the ICL loss of the linear transformer closely matches that of its softmax counterpart, as shown in Figure 14(b).

With the learned patterns, we have the multi-layer linear transformer iteration rules for $\ell \in [L] \cup \{q\}$:

$$\begin{aligned}
z_\ell^{(t+1)} &= z_\ell^{(t)} + \frac{1}{L}\sum_{h=1}^{H} \Psi^{(h)} Z_{1:L}^{(t)} \cdot (Z_{1:L}^{(t)\top} \Phi^{(h)} z_\ell^{(t)}) \\
&= \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \frac{1}{L}\sum_{h=1}^{H} \begin{bmatrix} u_1^{(h)} I_d & 0_d \\ 0_d^\top & u_2^{(h)} \end{bmatrix} \cdot \begin{bmatrix} X_{1:L}^{(t)} \\ Y_{1:L}^{(t)} \end{bmatrix} \cdot \begin{bmatrix} X_{1:L}^{(t)} \\ Y_{1:L}^{(t)} \end{bmatrix}^\top \cdot \begin{bmatrix} \omega^{(h)} I_d & 0_d \\ 0_d^\top & 0 \end{bmatrix} \cdot \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} \\
&= \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \frac{1}{L}\sum_{h=1}^{H} \begin{bmatrix} u_1^{(h)} X_{1:L}^{(t)} X_{1:L}^{(t)\top} & u_1^{(h)} X_{1:L}^{(t)} Y_{1:L}^{(t)\top} \\ u_2^{(h)} Y_{1:L}^{(t)} X_{1:L}^{(t)\top} & u_2^{(h)} Y_{1:L}^{(t)} Y_{1:L}^{(t)\top} \end{bmatrix} \cdot \begin{bmatrix} \omega^{(h)} x_\ell^{(t)} \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \frac{1}{L} \begin{bmatrix} \langle u_1, \omega \rangle X_{1:L}^{(t)} X_{1:L}^{(t)\top} x_\ell^{(t)} \\ \langle u_2, \omega \rangle Y_{1:L}^{(t)} X_{1:L}^{(t)\top} x_\ell^{(t)} \end{bmatrix} \\
&= \begin{bmatrix} x_\ell^{(t)} \\ y_\ell^{(t)} \end{bmatrix} + \frac{1}{L} \begin{bmatrix} \langle u_1, \omega \rangle \sum_{\ell'=1}^{L} x_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)} \\ \langle u_2, \omega \rangle \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} x_{\ell'}^{(t)\top} \cdot x_\ell^{(t)} \end{bmatrix},
\end{aligned} \tag{21}$$

where $Z_{1:L}^{(t)}$, $X_{1:L}^{(t)}$, $Y_{1:L}^{(t)}$ means $\{z_\ell^{(t)}\}_{\ell=1}^L$, $\{x_\ell^{(t)}\}_{\ell=1}^L$, and $\{y_\ell^{(t)}\}_{\ell=1}^L$ respectively. We find that the update rule learned by the multi-layer linear-attention transformer matches Algorithm 1, the same procedure identified for the softmax case in Corollary 1. As shown in Figures 15(a) and (b), the prediction loss of the linear transformer and that of its interpreted algorithm are nearly identical. In addition, the alignment between the two models is high. Together, these provide strong empirical support for our theoretical analysis. We observe similar behavior in non-looped linear transformers (Figure 16). However, the patterns in these non-looped models are inconsistent and depend on the initialization, similar to their softmax counterparts. The learned structural patterns confirm that the model implements Algorithm 1.

In addition, as reported in Table 1, the learned inner products $\langle u_1, \omega \rangle$ and $\langle u_2, \omega \rangle$ at each layer of the looped linear model are nearly identical to those of the looped softmax transformer. The learned step sizes η_x and η_y also match, suggesting that in the varying-covariance setting, trained linear transformers are functionally equivalent to softmax transformers for solving linear regression tasks.

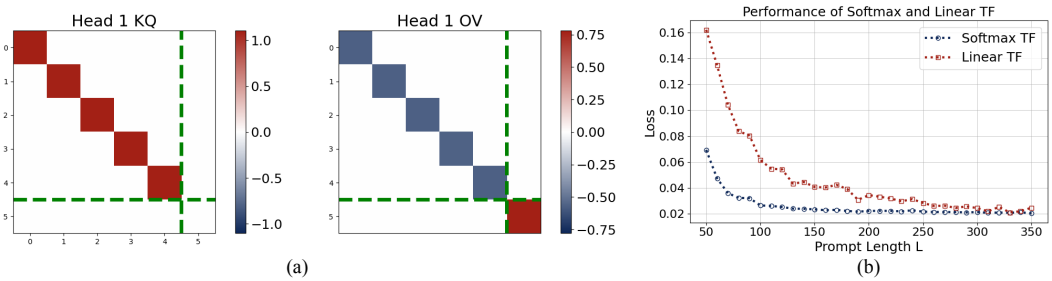


Figure 14: Results for a 2-layer, 1-head linear transformer: (a) KQ and OV pattern visualizations; (b) performance comparison with a softmax transformer of the same depth.

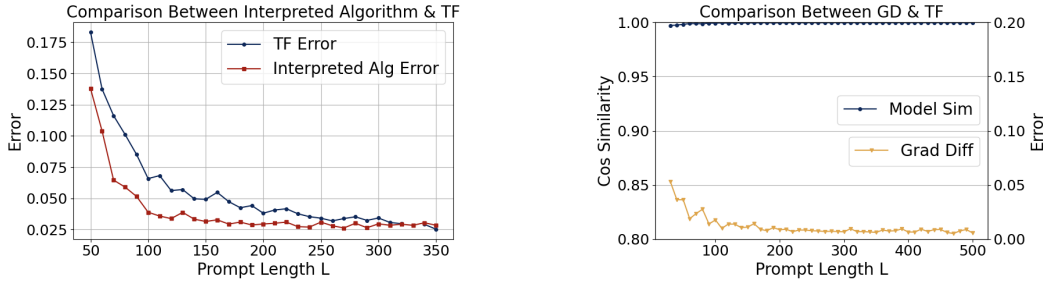


Figure 15: Results for a 2-layer 1-head linear transformer: (a) Performance comparison between the transformer and the interpreted algorithm. (b) Alignment between the transformer and the interpreted algorithm.

C.3 Varying Covariance vs. Constant Covariance

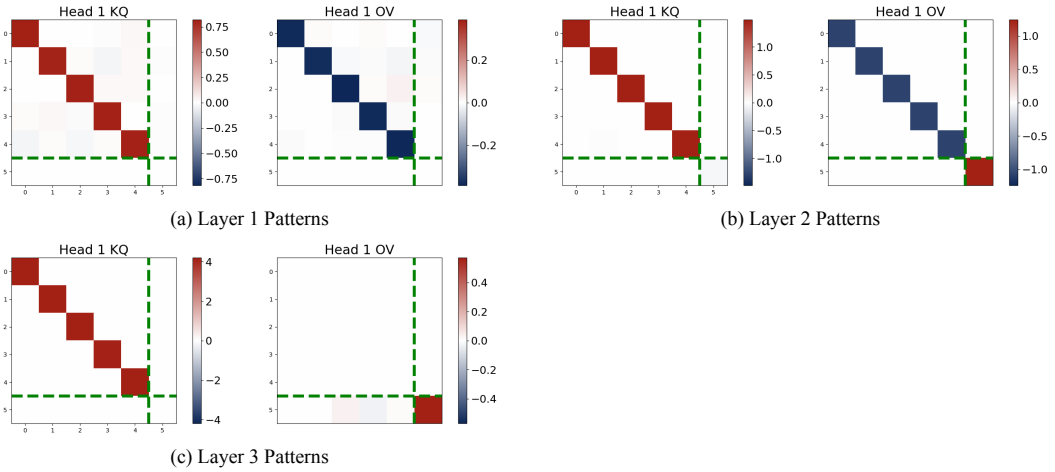
In this subsection, we analyze how transformers behave when trained on input distributions with constant covariance, a setting that extends the prior works (Ahn et al., 2023; Von Oswald et al., 2023) to softmax-attention models. In addition, we study the generalization ability of models trained with varying covariance when tested on a constant-covariance distribution.

Transformers Trained with Constant Covariance. We first consider the case in which covariates are sampled from $\mathcal{N}(0, \Sigma)$ with a covariance matrix Σ that is fixed for each sequence. Two settings are examined: (1) Σ is the identity matrix; and (2) Σ is a KMS matrix (Fikioris, 2018) with parameter $\rho = 0.5$, whose (i, j)-entry is $\Sigma_{ij} = \rho^{|i-j|}$. We train a 2-layer, 2-head looped transformer using the same experimental setup as in Section 4, and obtain the following observation.

Observation. Transformers encode the covariance structure into the KQ circuits when the covariance of x_ℓ is constant.

The learned patterns differ from those reported in Section 4 and exhibit a strong dependence on Σ. Figures 22(a) and (b) display the KQ and OV patterns for the two cases, respectively. When the covariates are anisotropic, the KQ patterns are no longer diagonal, although the OV patterns remain consistent with Equation equation 5. The learned structure can be expressed as

$$\Phi^{(h)} = \begin{bmatrix} \omega^{(h)} \Omega I_d & 0_d \\ 0_d^\top & 0 \end{bmatrix}, \qquad \Psi^{(h)} = \begin{bmatrix} u_1^{(h)} I_d & 0_d \\ 0_d^\top & u_2^{(h)} \end{bmatrix}, \tag{22}$$



(a) Layer 1 Patterns

(b) Layer 2 Patterns

(c) Layer 3 Patterns

Figure 16: Visualization for a 3-layer 1-head linear transformer with structured patterns.

Table 1: The learned inner products at each layer.

Looped Softmax Transformer			
	2 Layers	3 Layers	4 Layers
$\langle u_1, \omega \rangle$	-0.6675	-0.6128	-0.6590
$\langle u_2, \omega \rangle$	0.8591	0.7193	0.6563
Looped Linear Transformer			
	2 Layers	3 Layers	4 Layers
$\langle u_1, \omega \rangle$	-0.6616	-0.6122	-0.6466
$\langle u_2, \omega \rangle$	0.8479	0.7214	0.6217

where Ω is a $d \times d$ preconditioning matrix. Prior work (He et al., 2025) shows that in single-layer transformers with sufficiently long sequences, Ω approximates Σ^{-1}. Our experiments confirm that this result also holds in the multi-layer setting.

Figure 22(c) visualizes Σ^{-1}, revealing a close correspondence between Σ^{-1} and the learned KQ patterns. Figure 22(d) plots the L2 distance between $\Phi \cdot \Sigma$ and a scaled identity matrix. The distance decreases during training and converges to a small value, indicating strong alignment.

These findings suggest that the transformer implements an alternative update rule:

$$
\begin{aligned}
x_\ell^{(t+1)} &= x_\ell^{(t)} + \frac{\langle u_1, \omega \rangle}{L} \sum_{\ell'=1}^{L} x_{\ell'}^{(t)} {x_{\ell'}^{(t)}}^\top \Sigma^{-1} x_\ell^{(t)}, \\
y_\ell^{(t+1)} &= y_\ell^{(t)} + \frac{\langle u_2, \omega \rangle}{L} \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} {x_{\ell'}^{(t)}}^\top \Sigma^{-1} x_\ell^{(t)}.
\end{aligned}
\tag{23}
$$

Here, u_1 is defined as the mean of the diagonal elements of $\Phi \cdot \Sigma$, excluding the last entry.

Figure 22(e) compares the performance of the trained transformer and the interpreted algorithm. The covariance matrix of the input data distribution is a KMS matrix. The two curves nearly overlap, validating the correctness of our interpretation. We conclude that multi-layer transformers can effectively handle covariates with constant covariance by adapting the learned algorithm to incorporate the covariance structure, in a manner analogous to the single-layer case.

Depth Generalization of the Algorithm in the Constant-Covariance Regime. We also test the depth generalization of the algorithm in this regime; see Figure 17. We find that the algorithm is also non-convergent and that the learned parameters are highly correlated with the model's depth. This

observation is consistent with the findings in the main paper, suggesting that, regardless of the input distribution, a multi-layer transformer learns a non-convergent algorithm that approximates linear regression in a manner that depends strongly on its depth.

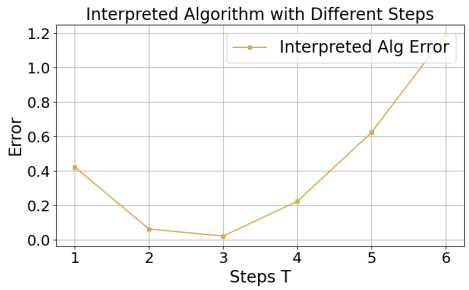


Figure 17: ICL error of the interpreted algorithm with varying number of steps.

Transformer Inference under Constant Covariance. We next examine how a transformer trained with varying covariance generalizes to test data with constant covariance. Specifically, we compare two multi-layer transformers: one trained with varying covariance and the other with constant covariance. The test data are sampled from a Gaussian distribution with identity covariance matrix.

As shown in Figure 22(e), both models perform well. The transformer trained with varying covariance achieves performance only slightly worse (by about 0.04) than the model trained directly with constant covariance. This demonstrates the flexibility of transformers across different input distributions. Since constant covariance is a special case of varying covariance, the model trained on the broader distribution retains strong generalization.

Discussion of Algorithmic Differences Between Our Work and Prior Work. Prior work (Von Oswald et al., 2023; Ahn et al., 2023) has analyzed the behavior of multi-layer linear transformers. Specifically, Ahn et al. (2023) studied inputs drawn from $\mathcal{N}(0, I_d)$ and concluded that the model learns a GD algorithm, while Von Oswald et al. (2023) extended this analysis to $\mathcal{N}(0, \Sigma)$ with a constant covariance matrix throughout training. A key feature of these results is that the interpreted algorithm is closely tied to the structure of the input covariance matrix, which is consistent with our conclusion under softmax attention in Equation 23. This correspondence arises because their training sequences are sampled from a single data distribution with a fixed covariance matrix Σ, allowing the transformer to encode this matrix as an inductive bias in its weights. As a result, a single-layer attention module suffices to solve the task by implementing a single preconditioned GD step.

We extend this analysis to a more challenging and realistic setting in which the covariance matrix Σ varies across sequences. In this setting, no fixed covariance exists, and the transformer must learn to adapt to changing covariance matrices on the fly. We find that the transformer learns a different algorithm to accomplish this, as detailed in the *interpreted algorithm* in the main paper. Rather than relying on any specific Σ in its formulation, this algorithm produces predictions that depend only on the current sequence and a depth-specific step size. Because the accuracy of the algorithm depends on the number of iterations, a single-layer transformer performs poorly, whereas increasing the number of layers leads to substantially better performance.

C.4 Single-layer Transformer vs. Multi-layer Transformer

In this subsection, we analyze the differences between the KQ and OV circuits learned by a single-layer transformer and those learned by a multi-layer transformer. As shown in Figure 18(a), when a single-layer model is applied to linear regression with varying covariance structures, the learned patterns exhibit slight differences. Specifically, the first d diagonal entries of the OV circuit are close to zero. Given the single-layer architecture, the last column of the KQ circuit and the first d columns of the OV circuit do not contribute to the final prediction $\hat{y}_q$. Similarly, the residual connection has no effect on the prediction because the last entry of the input is set to zero. Consequently, the single-layer

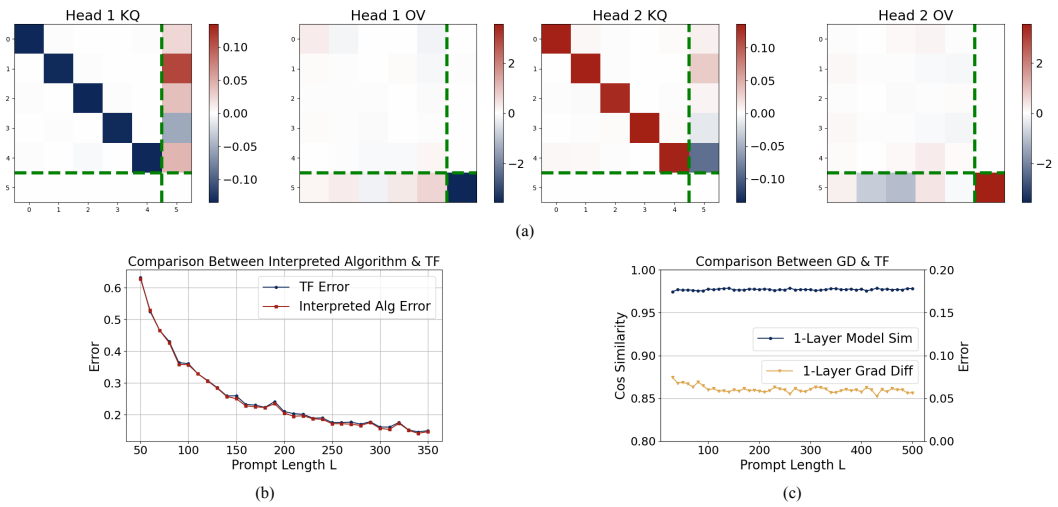


Figure 18: Results for a 1-layer, 2-head transformer: (a) visualizations of KQ and OV patterns; (b) performance comparison between the transformer and the interpreted algorithm; (c) alignment between the transformer and the interpreted algorithm.

transformer output can be expressed as

$$\hat{y}_q = \frac{\langle u_2, \omega \rangle}{L} \sum_{\ell'=1}^{L} y_{\ell'}^{(t)} {x_{\ell'}^{(t)}}^{\top} \cdot x_{\ell}^{(t)}. \tag{24}$$

This formulation reveals that the single-layer transformer learns an algorithm similar to Algorithm 1 but performs only a single iteration. This single-step limitation leads to its inferior performance. The empirical evidence in Figures 18(b) and (c) supports the close alignment between the interpreted algorithm and the transformer.

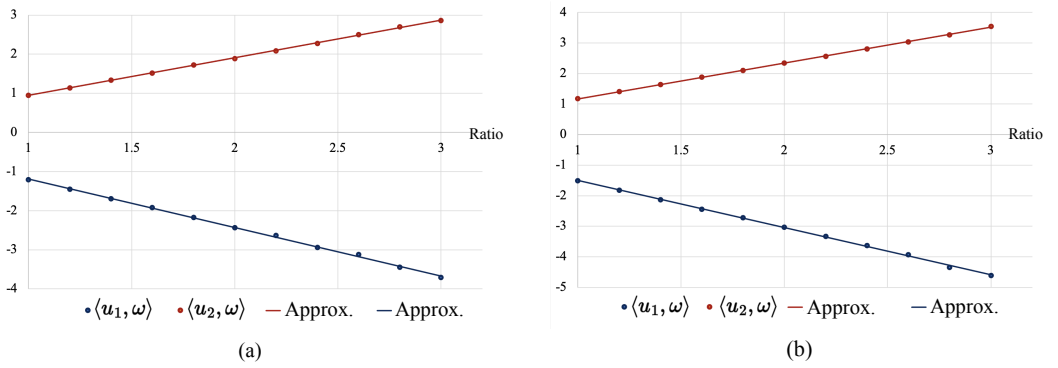


Figure 19: Results for relationship between covariance and inner products: (a) the approximation of the relationships $\langle u_1, \omega \rangle$ vs. eigenvalue ratios and $\langle u_2, \omega \rangle$ vs. eigenvalue ratios when the initial eigenvalues are constant; (b) the approximation relationship when initial eigenvalues are various.

C.5 Relationship Between the Covariance and the Inner Products

In this subsection, we investigate how the eigenvalues $(\lambda_1, \ldots, \lambda_d)$ of the covariance matrix influence the inner products $\langle u_1, \omega \rangle$ and $\langle u_2, \omega \rangle$. Here we only simply explore the trend of this influence, rather than exact relationship. We study two cases: a fixed set of eigenvalues and a varying set of

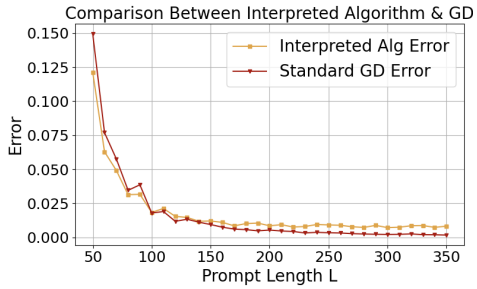


Figure 20: Performance comparison between interpreted algorithm and GD algorithm with the optimal step-size.

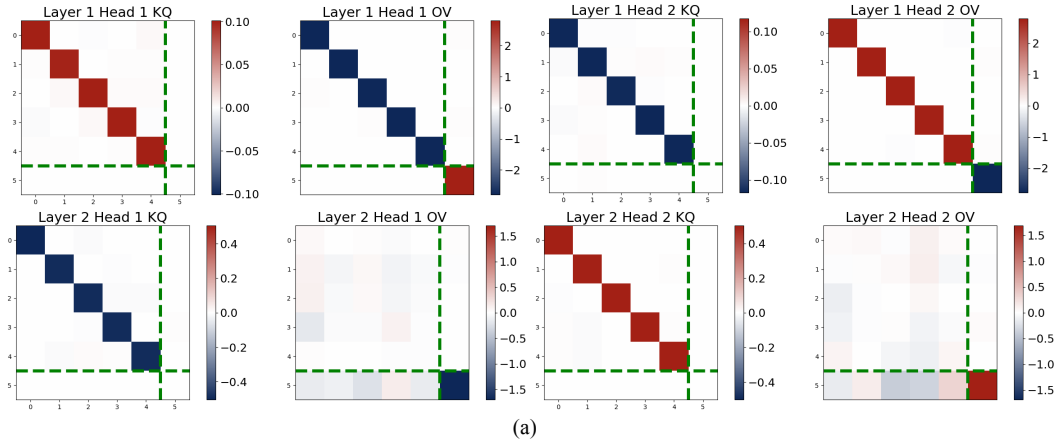
eigenvalues. In the first case, the reference eigenvalues are fixed as $\Lambda_0 = (1.0, 0.75, 0.5, 0.75, 1.0)$. We introduce a scaling factor k such that $\Lambda = k \cdot \Lambda_0$. In the second case, the eigenvalues are sampled from $\mathrm{Unif}(\lambda_{\min}, \lambda_{\max})$, with reference values $\lambda_{\min} = 0.8$ and $\lambda_{\max} = 1.2$. The scaling factor k is applied to both bounds. In both cases, we vary k from 1 to 3 in increments of 0.2 and conduct experiments using a two-layer transformer.

Across both settings, the KQ and OV circuits remain diagonal, consistent with the findings in Section 4. Examining the learned inner products as a function of the scaling factor k, we arrive at the following key result.

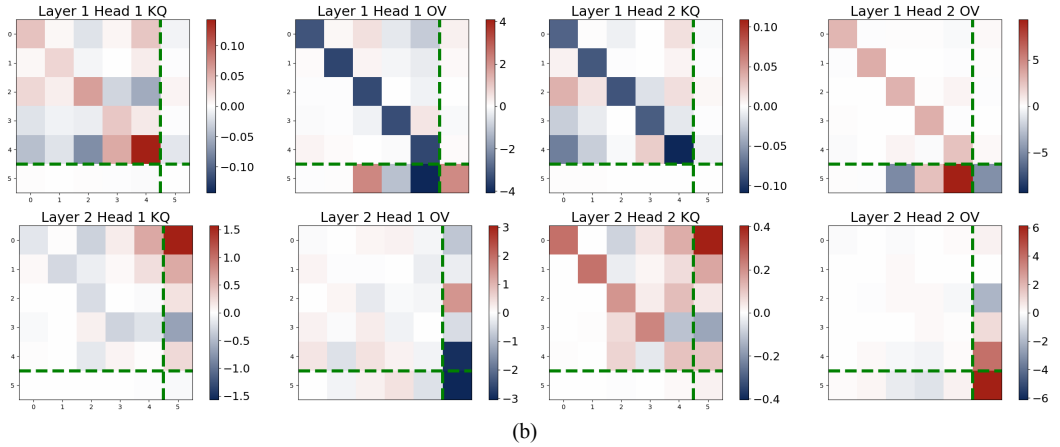
Observation. The inner products $\langle u_1, \omega \rangle$ and $\langle u_2, \omega \rangle$ are inversely proportional to the scaling factor k.

This observation holds in both the fixed- and varying-eigenvalue settings. As shown in Figure 19, the relationship between k and the reciprocals of the inner products is approximately linear, validating our observation.

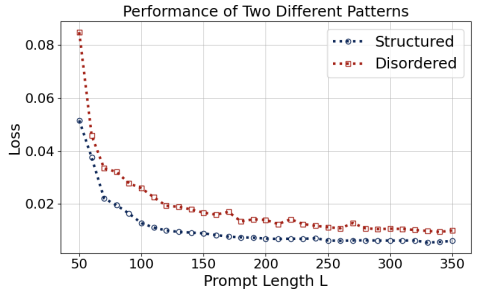
These results highlight a primary connection between the eigenvalues of the covariance matrix and the learned inner products. A deeper investigation into this relationship, including its theoretical underpinnings and implications for generalization, is an important direction for future work.



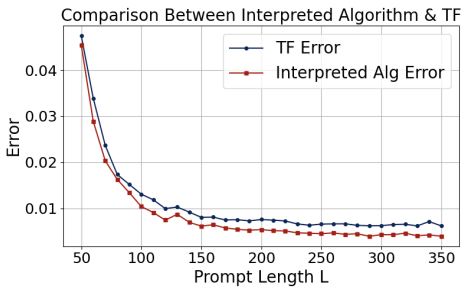
(a)



(b)



(c)



(d)

Figure 21: Comparison of two pattern types: (a) structured patterns in a 2-layer, 2-head transformer; (b) disordered patterns in the same architecture with different initialization; (c) ICL prediction loss for structured vs. disordered patterns; (d) performance of the structured-pattern transformer vs. its interpreted algorithm.

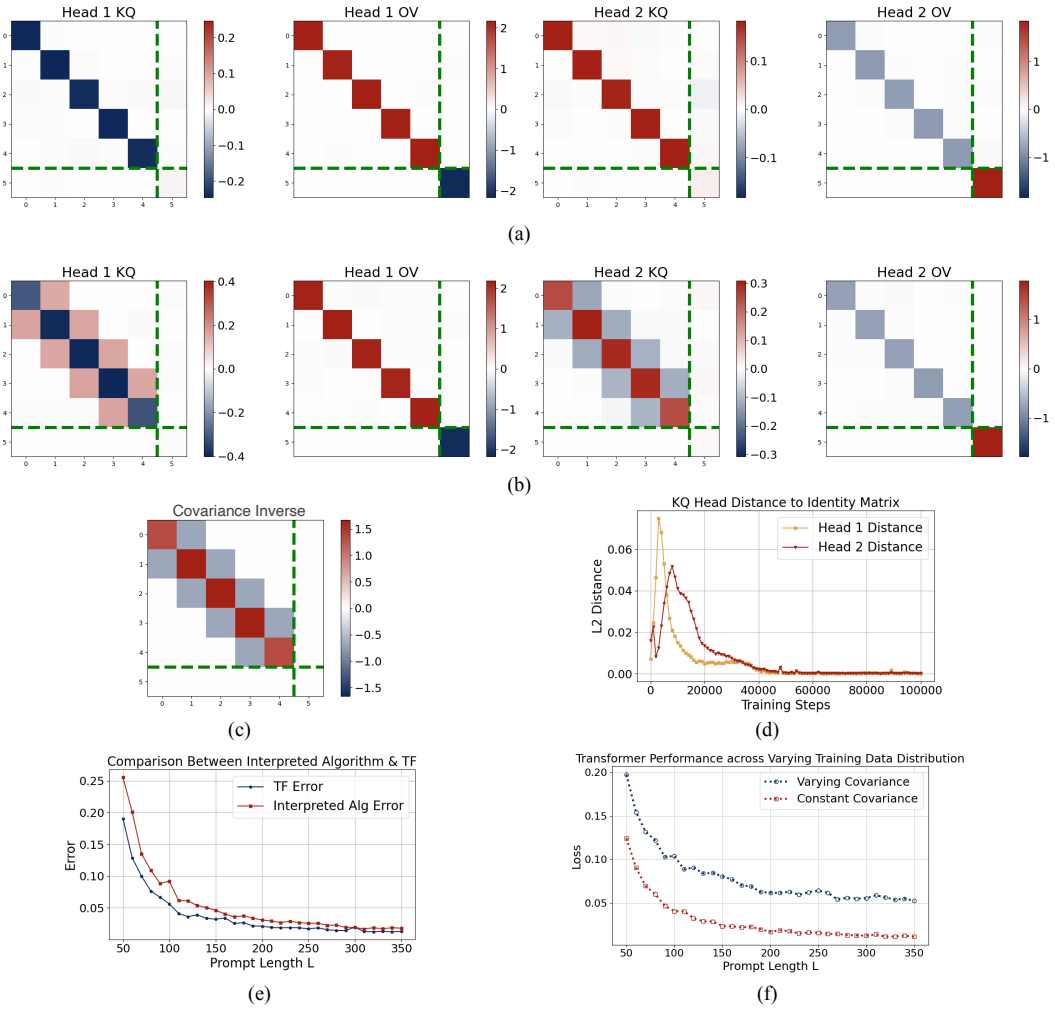


Figure 22: Results for transformers trained with different input distributions: (a) KQ and OV patterns for covariates from $\mathcal{N}(0, I_d)$; (b) KQ and OV patterns for covariates from $\mathcal{N}(0, \Sigma)$ with Σ a KMS matrix ($\rho = 0.5$); (c) visualizations of Σ^{-1}; (d) L2 distance between the $\Phi \cdot \Sigma^{-1}$ and $u_1 \cdot I_d$; (e) performance comparison between transformer and interpreted algorithm; (f) performance comparison between transformers trained on different input distributions.