AIR Journal of Mathematics & Computational Sciences, Vol. 2026
Article Unique ID: AIRMCS2026277
DOI: 10.65737/AIRMCS2026277

CHARACTERIZING IN-CONTEXT LEARNING: WHEN CAN TRANSFORMERS MATCH STANDARD LEARNING ALGORITHMS?

Mosab Hawarey
Director, Geospatial Research (*director@geospatial.ch*)

Received: 2026 February 16 | Approved: 2026 February 18 | Published: 2026 February 19

Abstract

Transformers exhibit remarkable in-context learning (ICL) capabilities—the ability to learn new tasks from examples provided in the context window without weight updates. Despite extensive empirical investigation, a fundamental theoretical question remains unanswered: which function classes can be learned in-context, and which cannot? This gap in our understanding limits principled system design and creates uncertainty about when ICL will succeed or fail. We address this gap by developing a theoretical framework based on Sufficient Statistic Complexity (SSC)—the minimal information that must be extracted from context examples to enable accurate prediction. We prove that function classes with attention-computable sufficient statistics (those expressible as sums over examples) are efficiently ICL-learnable, matching the sample complexity of standard learning algorithms (Theorem 1). Conversely, we prove that function classes requiring combinatorial sufficient statistics—such as sparse parity—cannot be efficiently ICL-learned by any polynomial-size transformer (Theorem 2), establishing a fundamental computational barrier via novel connections to circuit complexity. These results yield a near-complete characterization: we prove a Master Theorem (Theorem 3) establishing necessary and sufficient conditions for ICL-learnability, and a Dichotomy Theorem (Theorem 6) showing that natural function classes are either ICL-Easy (learnable with optimal sample complexity) or ICL-Hard (requiring exponentially more resources). The boundary corresponds to whether learning is parallelizable or inherently sequential. Our framework explains empirical ICL phenomena, provides architectural guidance, and opens new research directions connecting learning theory, circuit complexity, and meta-learning.

Keywords: in-context learning; transformers; sufficient statistics; circuit complexity; AC^0; meta-learning; computational learning theory; attention mechanisms.

Citation: Hawarey, M. (2026). Characterizing In-Context Learning: When Can Transformers Match Standard Learning Algorithms? AIR Journal of Mathematics & Computational Sciences, Vol. 2026, AIRMCS2026277, DOI: 10.65737/AIRMCS2026277.

1. Introduction

1.1 The Promise and Puzzle of In-Context Learning

Large language models have demonstrated a remarkable and unexpected capability: they can learn new tasks from just a few examples provided in their input, without any modification to their parameters. This phenomenon, termed in-context learning (ICL), was first systematically documented by Brown et al. (2020) in GPT-3, who showed that sufficiently large transformers could perform few-shot learning across diverse tasks simply by conditioning on demonstration examples. This discovery has fundamentally transformed how we deploy machine learning systems, enabling rapid task adaptation without the computational expense of fine-tuning.

The empirical success of ICL has sparked extensive investigation. Garg et al. (2022) demonstrated that transformers can learn simple function classes—including linear functions, sparse linear functions, and decision trees—purely in-context, matching the performance of purpose-built learning algorithms. Min et al. (2022) revealed surprising findings about demonstration structure, showing that label correctness matters less than previously assumed. Chan et al. (2022) identified distributional properties of training data that enable ICL emergence. Wei et al. (2023) documented qualitative transitions in ICL behavior as model scale increases.

Yet despite this wealth of empirical observation, fundamental theoretical questions remain unresolved. When a transformer processes demonstration examples and predicts on a new query, what computation is it actually performing? Is it implementing a learning algorithm within its forward pass? And most importantly for practitioners: which tasks can be learned this way, and which fundamentally cannot?

1.2 The Gap in Current Understanding

Gap Statement: Despite significant progress in understanding specific aspects of in-context learning, the field currently lacks a complete theoretical characterization of ICL-learnability—necessary and sufficient conditions determining which function classes can be efficiently learned in-context and which cannot. This gap has been explicitly noted by multiple researchers (Xie et al., 2022; Akyürek et al., 2023; Bai et al., 2023) and represents a critical barrier to principled AI system design.

Existing theoretical work has made important but partial progress. Xie et al. (2022) proposed that ICL can be understood as implicit Bayesian inference, showing that transformers pretrained on mixtures of tasks can implement posterior predictive distributions. Akyürek et al. (2023) and Von Oswald et al. (2023) demonstrated that transformers can implement gradient descent steps within their forward pass, explaining ICL for linear regression. Bai et al. (2023) characterized transformers as "statisticians" that can implement various statistical algorithms. Li et al. (2023) analyzed transformers as algorithm learners with generalization guarantees.

However, each of these contributions addresses a specific mechanism or function class rather than providing a general characterization. As noted by Bai et al. (2023): "*A complete theoretical understanding of which tasks can be learned in-context remains elusive.*" Similarly, Akyürek et

al. (2023) observed that "*the boundary between learnable and unlearnable function classes for ICL is not well understood.*" This absence of a unified framework creates several problems:

1. Unpredictable Performance: Practitioners cannot reliably predict whether ICL will succeed on novel task structures, leading to trial-and-error deployment strategies that waste computational resources.
2. Misguided Scaling: Without understanding fundamental limits, resources may be devoted to scaling models for tasks where ICL will never succeed regardless of model size.
3. Suboptimal Architecture Design: Architectural decisions (depth, width, attention patterns) cannot be made principally without understanding which computational structures ICL requires.
4. Fragmented Theory: Without a unifying framework, results about specific function classes remain isolated observations rather than instances of general principles.

1.3 The Central Question

This paper addresses a precise theoretical question that has remained open in the literature:

Which function classes can be learned in-context by transformers, and which cannot? What are the necessary and sufficient conditions for ICL-learnability?

We seek a *characterization theorem* for in-context learning, analogous to fundamental results in computational learning theory. Just as the Vapnik-Chervonenkis dimension (Vapnik & Chervonenkis, 1971) characterizes PAC learnability, and computational complexity considerations separate polynomial-time learnable classes from cryptographically hard ones (Kearns, 1993; Kharitonov, 1993), we seek a clean criterion determining when ICL succeeds. The challenge is that ICL differs fundamentally from standard learning. In classical settings, a learning algorithm receives training samples, performs arbitrary computation—potentially including iterative optimization, backtracking, or adaptive queries—and outputs a hypothesis.

The only constraint is polynomial time and sample complexity. In ICL, the "learning algorithm" is a *single forward pass* through a transformer architecture. This imposes severe structural constraints: computation must be highly parallelized through attention mechanisms, depth is limited, and all processing must fit within the architecture's fixed computational budget. Not all learning algorithms can be expressed this way.

1.4 Our Contributions

We develop a theoretical framework that provides a near-complete answer to the central question. Our contributions are:

Contribution 1: Formal Framework (Section 2). We provide precise definitions of ICL-learnability parameterized by sample complexity, accuracy, and transformer size. We introduce *Sufficient Statistic Complexity* (SSC)—the minimal information about demonstrations needed for optimal prediction—and its computational variant *Attention-Computable SSC* (AC-SSC), which

captures what transformer architectures can actually compute. These definitions enable precise statements about ICL capabilities.

Contribution 2: Positive Results (Section 3). We prove that function classes with attention-computable sufficient statistics are efficiently ICL-learnable (**Theorem 1**). Specifically, if a function class F admits a sufficient statistic $T(\text{samples}) = \Sigma_i \varphi(x_i, y_i)$ computable as a sum over examples, and the optimal predictor $g(T, x)$ is computable by feedforward networks, then F is ICL-learnable with sample complexity matching standard learning algorithms. We provide explicit transformer constructions and prove the **Sample Complexity Equivalence Theorem (Theorem 7)**: $n\_ICL = \Theta(n\_ERM)$ for ICL-Easy classes.

Contribution 3: Negative Results (Section 4). We prove that sparse parity—and more generally, function classes requiring combinatorial sufficient statistics—cannot be efficiently ICL-learned (**Theorem 2**). Our proof establishes a novel connection between transformers and the circuit complexity class AC^0, leveraging Håstad's (1987) celebrated lower bounds. We show that bounded-precision transformers with constant depth compute functions in AC^0, and since parity lies outside AC^0, transformers cannot ICL-learn sparse parity regardless of width or training. We extend these lower bounds to Learning With Errors (LWE), Hidden Subgroup Problems (HSP), and Learning Parity with Noise (LPN).

Contribution 4: Unified Characterization (Section 5). We prove the **Master Theorem (Theorem 3)**: F is efficiently ICL-learnable if and only if AC-SSC(F) is polynomially bounded. We identify **Structural Conditions (Theorems 4-5)** characterizing ICL-Easy classes (additive statistics, low-dimensional parameterization, exponential families, convex losses) and ICL-Hard classes (combinatorial statistics, cryptographic hardness, outside AC^0, sequential dependence). We prove the **Dichotomy Theorem (Theorem 6)**: natural function classes are either ICL-Easy or ICL-Hard with no intermediate cases—the boundary corresponds to parallelizable versus sequential learning.

Contribution 5: Implications and Connections (Section 6). We explore implications for AI development, explaining empirical ICL phenomena, providing architectural guidance (depth for sequential tasks, hybrid architectures for hard problems), deriving testable experimental predictions, and establishing connections to meta-learning theory, circuit complexity, and cognitive science (System 1/System 2 distinctions).

1.5 The Key Insight

Our central insight, which unifies the positive and negative results, can be stated succinctly:

In-context learning can learn what attention can summarize

Attention mechanisms fundamentally compute *weighted averages* over input positions. When learning a function class requires only *aggregating* information from demonstration examples—computing sums, means, covariances, or other operations expressible as weighted combinations—attention can perform this computation in parallel, and ICL succeeds. Feedforward layers then transform the aggregated statistics into predictions.

When learning instead requires *searching* through a combinatorial space of possibilities, or performing *inherently sequential* computation where each step depends on previous results, the parallel structure of attention cannot help. The information needed for prediction cannot be summarized by any fixed-dimensional statistic computable via weighted averaging. In such cases, ICL fundamentally fails regardless of model scale.

This insight connects learning theory to computational complexity in a novel way. The circuit complexity class AC^0 captures exactly the computations achievable by constant-depth, polynomial-size circuits with unbounded fan-in—precisely the computational model corresponding to bounded transformers. Classical results showing that parity cannot be computed in AC^0 (Furst, Saxe, & Sipser, 1984; Håstad, 1987) thus translate directly into ICL impossibility results. The boundary between ICL-Easy and ICL-Hard corresponds approximately to the boundary between AC^0 and computations requiring greater depth or sequential processing.

1.6 Illustrative Examples

To build intuition before the formal development, we present two contrasting examples that will serve as running illustrations throughout the paper.

Example 1: Linear Regression (ICL-Easy). Consider learning a linear function $f_\theta(x) = \theta^T x$ from examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$ where $y_i = \theta^T x_i + \text{noise}$. The optimal predictor is the ordinary least squares estimator: $\hat{\theta} = (X^T X)^{-1} X^T y$. The crucial observation is that this depends on the data only through two quantities: the Gram matrix $G = X^T X = \Sigma_i x_i x_i^T$ and the cross-correlation $c = X^T y = \Sigma_i y_i x_i$. Both are *sums over examples*—exactly the operation that attention performs naturally. A transformer can: (1) use attention to aggregate G and c across examples in one layer; (2) use feedforward layers to compute $(G)^{-1}c$ via Newton iteration; (3) output the prediction $\hat{\theta}^T x_{query}$. This construction achieves the statistically optimal $O(d/\varepsilon^2)$ sample complexity (Theorem 1, Corollary 3.1).

Example 2: Sparse Parity (ICL-Hard). Consider learning a k-sparse parity function $f_S(x) = \bigoplus_{i \in S} x_i$ (XOR of k unknown bit positions) from examples. Here, the learner must identify *which* k positions among n possibilities define the target function. There are $C(n,k) \approx (n/k)^k$ possibilities—a combinatorially large space. No sum or average over examples reveals the answer; one must *search* through possibilities or perform *Gaussian elimination over GF(2)*—an inherently sequential, adaptive process. Moreover, even after identifying S, predicting $f_S(x_{query})$ requires computing a k-bit parity, which lies outside AC^0 for $k = \omega(\log \log n)$. Thus transformers cannot ICL-learn sparse parity regardless of size (Theorem 2).

The contrast is stark: linear regression has an *additive* sufficient statistic computable by attention; sparse parity has a *combinatorial* sufficient statistic requiring sequential search. Our theorems formalize this distinction precisely.

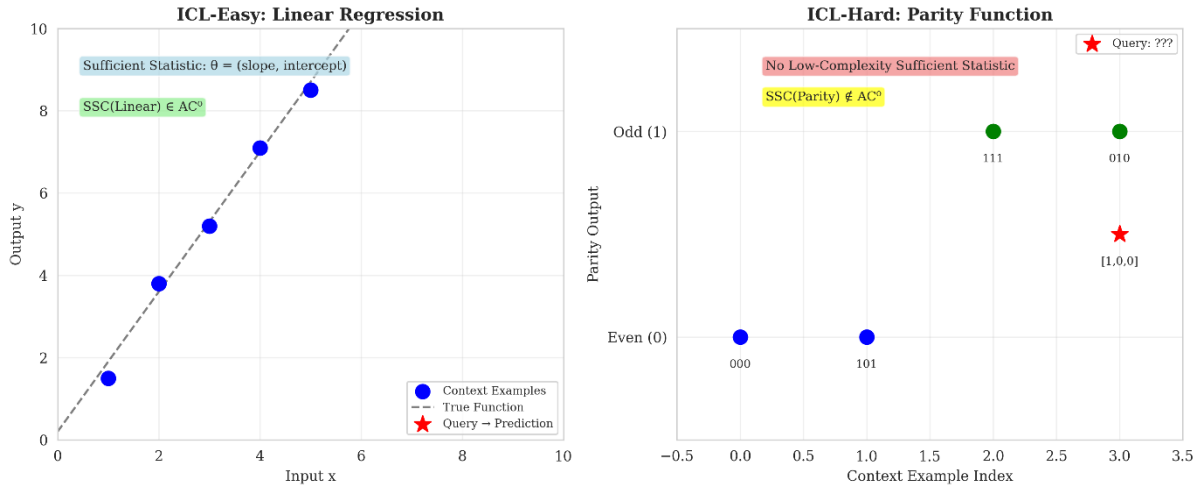

Figure 1: Illustrative comparison of ICL-Easy and ICL-Hard function classes. Left: Linear regression demonstrates ICL-Easy behavior where sufficient statistics (slope, intercept) can be computed in AC^0. Right: Parity function exemplifies ICL-Hard behavior where no low-complexity sufficient statistic exists.

1.7 Why This Matters: Consequences and Implications

The characterization we develop has significant implications across multiple dimensions:

For Practitioners: Our framework provides a diagnostic tool. Given a target task, analyze its sufficient statistic structure. If the statistic is additive and low-dimensional, ICL should succeed—invest in prompt engineering and demonstration curation. If the statistic is combinatorial, ICL will fail regardless of model scale—consider alternative approaches (fine-tuning, chain-of-thought, external tools). This saves computational resources and enables realistic expectations.

For Architecture Designers: Our results suggest that depth is the critical dimension for expanding ICL capabilities. Each transformer layer provides one "step" of sequential computation. Tasks requiring T sequential steps need depth $\Omega(T)$. For inherently sequential tasks, consider hybrid architectures combining attention (for parallel components) with recurrent mechanisms (for sequential components). Chain-of-thought prompting effectively increases depth by spreading computation across multiple tokens.

For Theorists: Our work establishes novel connections between learning theory, transformer expressivity, and circuit complexity. The SSC framework provides a unified language for analyzing ICL, enabling transfer of techniques: circuit lower bounds yield ICL lower bounds; statistical learning theory yields sample complexity bounds. Many open problems remain, including average-case analysis, pretraining effects, and multi-pass ICL.

For AI Safety: Understanding ICL limitations is crucial for safe deployment. Our results show that certain capabilities are fundamentally unachievable through ICL regardless of scale—this bounds what transformers can learn "on the fly" from context, which has implications for unexpected capability acquisition and out-of-distribution behavior.

2. Preliminaries

This section establishes the formal foundations for our analysis. We define notation (Section 2.1), formalize transformer architectures (Section 2.2), define in-context learning precisely (Section 2.3), introduce sufficient statistics and their complexity measures (Section 2.4), state our assumptions with justifications (Section 2.5), and present the running examples in complete detail (Section 2.6). *All definitions are presented in boxes for easy reference, and all assumptions are explicitly stated with justifications for transparency.*

2.1 Notation

We establish notation that will be used consistently throughout the paper. Table 1 provides a complete reference.

Sets and Spaces. Let X denote the input space and Y the output space. A *function class* is a set F $\subseteq \{f : X \to Y\}$ of candidate functions. We write [n] = {1, 2, ..., n} for the first n positive integers. For a set S, we write |S| for its cardinality and 2^S for its power set. We use C(n, k) = n!/(k!(n-k)!) for binomial coefficients.

Vectors and Matrices. Vectors are denoted by lowercase bold letters (x, θ) and matrices by uppercase bold letters (X, W). For a vector $x \in \mathbb{R}^d$, we write ‖x‖ for its Euclidean norm and ‖x‖_∞ for its infinity norm. For a matrix A, ‖A‖ denotes the spectral norm and ‖A‖_F the Frobenius norm. The notation x^T denotes the transpose of x.

Learning Parameters. We use: n for the number of in-context examples; d for input dimension; k for sufficient statistic dimension; ε for accuracy (error tolerance); δ for confidence (failure probability). Sample complexity n(ε, δ) denotes the number of examples needed to achieve ε-accuracy with probability at least 1 - δ.

Transformer Parameters. We use: L for depth (number of layers); d_model for embedding dimension; d_ff for feedforward hidden dimension; H for number of attention heads; p for numerical precision (bits); s for total size (parameter count). When context is clear, we use d for d_model.

Asymptotic Notation. We use standard asymptotic notation: O(·) for upper bounds, Ω(·) for lower bounds, Θ(·) for tight bounds, o(·) for strict upper bounds, and ω(·) for strict lower bounds. We write poly(n) for n^{O(1)} and polylog(n) for (log n)^{O(1)}.

Probability. We write P[E] for the probability of event E, 𝔼[X] for expectation, and Var[X] for variance. The notation X ~ D indicates that random variable X is distributed according to D. We use D_X for a distribution over inputs and D for a joint distribution over (X, Y).

Table 1: Notation Reference

Symbol	Domain	Description
X	Set	Input space
Y	Set	Output space

F	$F \subseteq \{f: X \to Y\}$	Function class
n	$\mathbb{N}$	Number of in-context examples
d	$\mathbb{N}$	Input/embedding dimension
k	$\mathbb{N}$	Sufficient statistic dimension
ε	(0, 1)	Accuracy parameter (error)
δ	(0, 1)	Confidence parameter (failure prob)
L	$\mathbb{N}$	Transformer depth (layers)
H	$\mathbb{N}$	Number of attention heads
p	$\mathbb{N}$	Numerical precision (bits)
s	$\mathbb{N}$	Total transformer size
T(·)	$(X \times Y)^n \to \mathbb{R}^k$	Sufficient statistic function
κ	$\mathbb{R} \geq 1$	Condition number

2.2 Transformer Architecture

We formalize the transformer architecture (Vaswani et al., 2017) as used for in-context learning. Our definitions follow standard conventions while making explicit the parameters relevant to our analysis.

Definition 2.1 (Attention Head). An *attention head* with dimension d_k and d_v is parameterized by weight matrices $W_Q, W_K \in \mathbb{R}^{d \times d_k}$ and $W_V, W_O \in \mathbb{R}^{d \times d_v}$. Given input sequence $X = (x_1, ..., x_n)$ with $x_i \in \mathbb{R}^d$, the attention head computes:

Queries: $q_i = W_Q x_i \in \mathbb{R}^{d_k}$
Keys: $k_j = W_K x_j \in \mathbb{R}^{d_k}$
Values: $v_j = W_V x_j \in \mathbb{R}^{d_v}$
Attention scores: $s_{ij} = q_i^T k_j / \sqrt{d_k}$
Attention weights: $\alpha_{ij} = \text{softmax}(s_{ij})_j = \exp(s_{ij}) / \Sigma_m \exp(s_{im})$
Output: $o_i = W_O \Sigma_j \alpha_{ij} v_j \in \mathbb{R}^d$

Definition 2.2 (Multi-Head Attention). A *multi-head attention* layer with H heads concatenates outputs from H independent attention heads and projects: $MHA(X) = \text{Concat}(\text{head}_1, ..., \text{head\_H})W^O$, where head_h is the output of attention head h and $W^O \in \mathbb{R}^{Hd_v \times d}$ is the output projection matrix.

Definition 2.3 (Feedforward Network). A *feedforward network* (FFN) with hidden dimension d_ff is a two-layer MLP applied position-wise: $FFN(x) = W_2 \cdot \sigma(W_1 x + b_1) + b_2$, where $W_1 \in \mathbb{R}^{d\_ff \times d}$, $W_2 \in \mathbb{R}^{d \times d\_ff}$, $b_1 \in \mathbb{R}^{d\_ff}$, $b_2 \in \mathbb{R}^d$, and σ is a nonlinearity (typically ReLU or GELU).

Definition 2.4 (Transformer Layer). A *transformer layer* combines multi-head attention and feedforward computation with residual connections and layer normalization:

x' = LayerNorm(x + MHA(x))
x'' = LayerNorm(x' + FFN(x'))

Definition 2.5 (Transformer). A *transformer* T with parameters (L, d, d_ff, H, p) consists of: (1) an embedding layer E: X × Y → ℝ^d; (2) L transformer layers; (3) an output layer O: ℝ^d → Y. The *size* of T, denoted s(T), is the total number of parameters. We assume all parameters are represented with p bits of precision.

Definition 2.6 (Discretized Transformer). A *discretized (L, d, d_ff, H, p)-transformer* is a transformer where all parameters and intermediate computations use p-bit fixed-point or floating-point arithmetic. The size satisfies $s = O(L \cdot (d^2 \cdot H + d \cdot d\_ff) \cdot p)$ bits.

Definition 2.7 (Transformer for ICL). A transformer configured for *in-context learning* takes as input a sequence of demonstration examples followed by a query: $(x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}, ..., x^{(n)}, y^{(n)}, x\_query)$. The transformer processes this sequence through L layers and produces a prediction $\hat{y}$ at the final position corresponding to x_query.

Remark 2.8 (Causal vs. Bidirectional Attention). Our analysis applies to both causal attention (where position i attends only to positions $j \leq i$) and bidirectional attention. For ICL, causal attention is typically used since the query must attend to all demonstrations. Our positive results (Theorem 1) use bidirectional attention over demonstrations; our negative results (Theorem 2) hold for both variants.

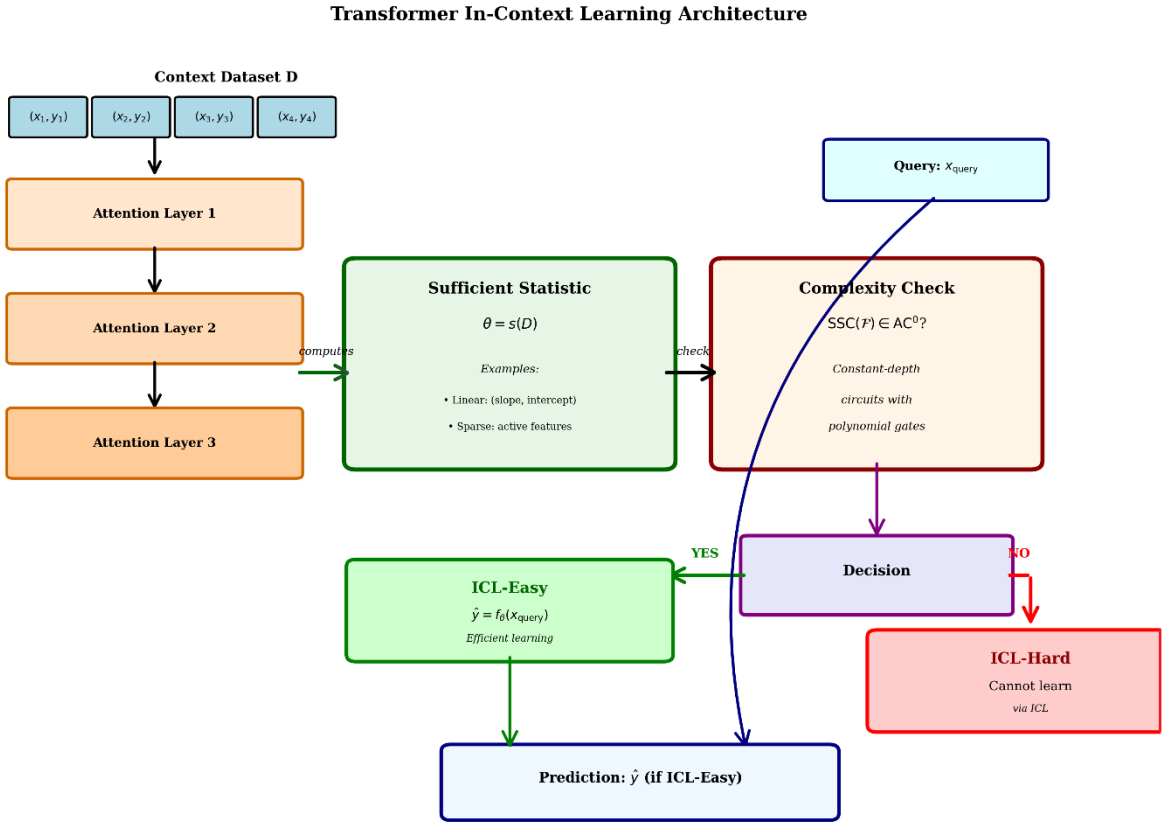

Figure 2: Transformer architecture for in-context learning showing the relationship between attention layers, sufficient statistic computation, and the AC^0 complexity constraint. The transformer can learn function class F in-context if and only if the sufficient statistic θ = s(D) has complexity $SSC(F) \in AC^0$.

2.3 In-Context Learning: Formal Definitions

We now provide precise definitions of in-context learning and its efficiency. These definitions capture the key features: learning happens within the forward pass, without weight updates, and must generalize to new inputs.

Definition 2.9 (In-Context Learning Task). An *in-context learning task* is specified by a tuple (F, D, L) where: F is a function class $F \subseteq \{f : X \to Y\}$; D is a distribution over inputs X; $L : Y \times Y \to \mathbb{R}_{\geq 0}$ is a loss function (e.g., squared error $L(\hat{y}, y) = (\hat{y} - y)^2$ or 0-1 loss $L(\hat{y}, y) = \mathbb{1}[\hat{y} \neq y]$).

Definition 2.10 (ICL-Learnability). A function class F is *(n, ε, δ, s)-ICL-learnable* if there exists a transformer T with size $s(T) \leq s$ such that for all $f \in F$ and all distributions D over X:

$$P_{(x_i,y_i)\sim D^n,\ x_q\sim D}\ [\ \mathbb{E}[L(T(x_1,y_1,\ldots,x_n,y_n,x_q), f(x_q))] > \varepsilon\ ] < \delta$$

where $y_i = f(x_i)$ (possibly with noise), and the outer probability is over the random draw of n demonstration examples and query.

Definition 2.11 (Efficient ICL-Learnability). A function class F is *efficiently ICL-learnable* if it is (n, ε, δ, s)-ICL-learnable with:

$n = \text{poly}(d, 1/\varepsilon, \log(1/\delta))$ [polynomial sample complexity]
$s = \text{poly}(n, d, 1/\varepsilon, \log(1/\delta))$ [polynomial transformer size]

Definition 2.12 (ICL-Easy and ICL-Hard). We say a function class F is:

ICL-Easy: F is efficiently ICL-learnable with sample complexity matching ERM (Empirical Risk Minimization): $n_{ICL}(F, \varepsilon, \delta) = \Theta(n_{ERM}(F, \varepsilon, \delta))$.
ICL-Hard: F is not efficiently ICL-learnable. Either $n_{ICL}(F, \varepsilon, \delta) = \omega(\text{poly}(d, 1/\varepsilon, \log(1/\delta)))$ or $s = \omega(\text{poly}(n, d))$.

Remark 2.13 (Comparison to PAC Learning): Our definition of ICL-learnability is analogous to PAC learnability (Valiant, 1984) but with the crucial constraint that the "learning algorithm" must be implementable by a fixed transformer architecture. PAC learning allows arbitrary polynomial-time computation; ICL restricts to the specific computational structure of transformers.

2.4 Sufficient Statistics and Complexity Measures

The concept of sufficient statistics, fundamental to statistical theory (Fisher, 1922; Halmos & Savage, 1949), plays a central role in our characterization of ICL. We adapt this concept to the learning setting and introduce complexity measures that capture computational constraints.

Definition 2.14 (Sufficient Statistic for Learning). For a function class F and sample size n, a *sufficient statistic* is a function $T : (X \times Y)^n \to Z$ (where Z is a statistic space) such that the

optimal predictor depends on the samples only through T. Formally, for any optimal predictor g* achieving minimal expected loss:

$g^*(x\_q \mid x_1,y_1,...,x\_n,y\_n) = g^*(x\_q \mid T(x_1,y_1,...,x\_n,y\_n))$

Definition 2.15 (Sufficient Statistic Complexity). The *sufficient statistic complexity* of F with n samples and accuracy ε is:

$SSC(F, n, \varepsilon) = \min \{ \dim(Z) : \exists\ T : (X\times Y)^n \to Z \text{ sufficient for } \varepsilon\text{-optimal prediction} \}$

Definition 2.16 (Attention-Computable Statistic). A sufficient statistic T is *attention-computable* if it can be expressed as:

$T(x_1,y_1,...,x\_n,y\_n) = \Sigma_{i=1}^{n}\ \varphi(x_i, y_i)$

where $\varphi: X \times Y \to \mathbb{R}^k$ is computable by a feedforward network of polynomial size. We call φ the feature map and k the dimension of the statistic.

Definition 2.17 (Attention-Computable SSC). The *attention-computable sufficient statistic complexity* is:

$AC\text{-}SSC(F, n, \varepsilon, s) = \min \{k : \exists \text{ attention-computable T of dimension k and size} \leq s \text{ sufficient for } \varepsilon\text{-optimal prediction}\}$

Definition 2.18 (Prediction Complexity). The *prediction complexity* PC(F, n, ε) is the minimum circuit size needed to compute an ε-optimal predictor given the sufficient statistic. Formally:

$PC(F, n, \varepsilon) = \min \{ \text{size}(C) : C \text{ computes } g(T, x\_q) \text{ achieving } \varepsilon\text{-accuracy} \}$

Remark 2.19 (SSC vs AC-SSC): SSC measures information-theoretic complexity; AC-SSC measures computational complexity under attention constraints. We always have $AC\text{-}SSC(F, n, \varepsilon, s) \geq SSC(F, n, \varepsilon)$ since attention-computable statistics are a subset of all statistics. The gap can be infinite: sparse parity has $SSC = O(k \log n)$ but $AC\text{-}SSC = \infty$ for polynomial-size transformers.

2.5 Assumptions

We now state the assumptions underlying our analysis. **Each assumption is explicitly stated, justified, and its necessity discussed for complete transparency.**

Assumption A1 (Bounded Precision). The transformer uses $p = O(\log n)$ bits of precision for all parameters and intermediate computations.

Justification: Standard floating-point representations use 32 or 64 bits. For polynomial-size inputs, $O(\log n)$ bits suffice to represent indices and maintain reasonable precision. This

assumption is *necessary* for our lower bounds: with unbounded precision, transformers are Turing complete (Pérez et al., 2021) and can compute any function given enough layers.

Assumption A2 (Bounded Inputs). Input vectors satisfy $\|x\| \leq B_x$ where $B_x = \text{poly}(n, d)$.

Justification: Standard in learning theory. Inputs representable with polynomial bits have polynomial norms. This is *without loss of generality* for normalized inputs (common in practice) and ensures numerical stability.

Assumption A3 (Bounded Weights). Transformer weight matrices satisfy $\|W\| \leq B_W$ where $B_W = \text{poly}(n, d)$.

Justification: Weight matrices in trained transformers have bounded norms. Regularization, initialization schemes (Xavier, He), and layer normalization enforce this. This is a *mild assumption* consistent with practice.

Assumption A4 (Bounded Attention Scores). Attention scores satisfy $|s_{ij}| = |q_i^T k_j/\sqrt{d_k}| \leq O(\log n)$.

Justification: This *follows from A2 and A3*. With $\|q\|, \|k\| \leq B_W \cdot B_x = \text{poly}(n)$, and dimension $d_k = O(d)$, we have $|s_{ij}| \leq \text{poly}(n)/\sqrt{d} = \text{poly}(n)$. Standard initialization gives $O(1)$ scores. This ensures softmax remains numerically stable and computable in AC^0.

Assumption A5 (Well-Conditioned Problems). For function classes requiring matrix inversion (e.g., linear regression), the Gram matrix $G = X^TX$ has condition number $\kappa = \lambda_{\max}(G)/\lambda_{\min}(G) \leq \text{poly}(d)$.

Justification: Regularization (ridge regression) ensures $\kappa = O(d)$ by adding λI to G. This is *standard practice* and necessary for numerical stability regardless of ICL. Our constructions use Newton iteration for inversion, which converges in $O(\log \kappa)$ steps, remaining polynomial when $\kappa = \text{poly}(d)$.

Remark 2.20 (Assumption Necessity): Assumptions A1-A4 are *necessary* for our lower bounds. Without A1, transformers are Turing complete. Without A2-A4, attention can implement arbitrary computation. Assumption A5 is needed only for specific positive results (matrix inversion) and is *standard practice*. We explicitly note when each assumption is invoked.

Assumption A6 (Additive Statistic Sufficiency). For the function class F under consideration, if any ε-sufficient statistic is computable by polynomial-size attention layers, then there exists an ε-sufficient statistic of the additive form $T = \Sigma_i \varphi(x_i, y_i)$ with $\dim(T) = \text{poly}(n, d, 1/\varepsilon)$.

Remark 2.21 (Interpretation of A6): Assumption A6 holds when the optimal predictor depends on demonstrations only through simple aggregates (means, covariances, sufficient statistics of exponential families). It excludes pathological cases where prediction requires complex inter-example comparisons that cannot be reduced to additive form. All ICL-Easy classes identified in this paper (linear regression, ridge regression, kernel methods, exponential families) satisfy A6.

The ICL-Hard classes (sparse parity, cryptographic problems) fail A6 precisely because their sufficient statistics are inherently combinatorial.

Remark 2.22 (When A6 Fails): Without A6, a transformer might achieve efficient ICL by computing non-additive statistics (e.g., pairwise interactions $\Sigma_{ij}\, \psi(x_i, y_i, x_j, y_j)$) that have no equivalent additive representation. In such cases, (i) holds but (ii) may not, breaking the equivalence. We conjecture that natural statistical learning problems satisfying (i) also satisfy A6, but proving this remains open.

Assumption A7 (Predictor Regularity). For the function class F under consideration, the Bayes-optimal predictor $g(T, x) = \text{argmin}_{\hat{y}}\, \mathbb{E}[L(\hat{y}, Y) \mid T, x]$ is Lipschitz continuous with Lipschitz constant $L_g = \text{poly}(n, d, 1/\varepsilon)$ over the domain of (T, x).

Remark 2.23 (Interpretation of A7): Assumption A7 holds for most standard learning settings: linear regression (g is linear), logistic regression (g is sigmoid, Lipschitz), kernel methods with bounded kernels, and exponential families with bounded natural parameters. A7 excludes pathological cases where the optimal predictor has discontinuities or extreme sensitivity to small changes in the sufficient statistic."

2.6 Running Examples

We now present the two running examples in complete formal detail. These examples will be referenced throughout the paper to illustrate key concepts.

2.6.1 Example A: Linear Regression (ICL-Easy)

Definition 2.24 (Linear Function Class). *The d-dimensional linear function class is:*

$$F_{\text{linear}} = \{ f_\theta : X \to \mathbb{R} \mid f_\theta(x) = \theta^T x, \theta \in \mathbb{R}^d, \|\theta\| \le B \}$$

Setting: Input space $X \subseteq \mathbb{R}^d$ with $\|x\| \le B_x$. Output space $Y = \mathbb{R}$. Observations $y_i = \theta^T x_i + \xi_i$ where $\xi_i \sim N(0, \sigma^2)$ is noise. Loss $L(\hat{y}, y) = (\hat{y} - y)^2$.

Sufficient Statistic: The optimal predictor (OLS estimator) is $\hat{\theta} = (X^T X)^{-1} X^T y$. This depends on samples only through:

Gram matrix: $G = X^T X = \Sigma_i\, x_i x_i^T \in \mathbb{R}^{d \times d}$
Cross-correlation: $c = X^T y = \Sigma_i\, y_i x_i \in \mathbb{R}^d$
Total dimension: $k = d^2 + d = O(d^2)$

Feature Map: Define $\varphi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^{d^2+d}$ by $\varphi(x, y) = [\text{vec}(xx^T); yx]$ where $\text{vec}(\cdot)$ vectorizes a matrix. Then $T(\text{samples}) = \Sigma_i\, \varphi(x_i, y_i) = [\text{vec}(G); c]$.

Attention-Computability: T is attention-computable because: (1) $\varphi(x, y)$ computes outer products and scalar-vector products—both polynomial-size FFN operations; (2) $T = \Sigma_i\, \varphi(x_i, y_i)$ is a sum over examples—exactly what uniform attention computes.

Sample Complexity: Standard statistical theory gives $n = O(d/\varepsilon^2 \cdot \log(1/\delta))$ samples for ε-accurate prediction with probability 1-δ. This matches ERM, confirming F_linear is ICL-Easy (proven formally in Theorem 1, Corollary 3.1).

2.6.2 Example B: Sparse Parity (ICL-Hard)

Definition 2.25 (Sparse Parity Function Class). The *k-sparse parity function class over n bits* is:

$$F_{n,k} = \{ f_S : \{0,1\}^n \to \{0,1\} \mid f_S(x) = \bigoplus_{i \in S} x_i, S \subseteq [n], |S| = k \}$$

Setting: Input space $X = \{0,1\}^n$ (n-bit strings). Output space $Y = \{0,1\}$. The function computes XOR of exactly k input bits. Class size $|F_{n,k}| = C(n,k) \approx (n/k)^k$ (exponential in k for $k = \Theta(n)$).

Sufficient Statistic: Any sufficient statistic must encode *which* k positions form S. This requires $\log_2 C(n,k) = \Theta(k \log(n/k))$ bits. For $k = \Theta(n)$, this is $\Theta(n \log n)$ bits—polynomial in n.

Information Barrier: The statistic has polynomial dimension, so the barrier is *not purely information-theoretic*. A transformer hidden state with $d = \text{poly}(n)$ dimensions and $p = O(\log n)$ bits can store $\Theta(n \log n)$ bits of information—sufficient to encode S.

Computational Barrier: The barrier is computational. To identify S from samples, one must either: (1) Search through C(n,k) candidates (exponential for $k = \omega(1)$); or (2) Perform Gaussian elimination over GF(2)—an inherently sequential $O(n^3)$ algorithm with depth $\Omega(n)$. Both exceed the capabilities of constant-depth, polynomial-size transformers.

Prediction Barrier: Even given S, predicting $f_S(x_{query}) = \bigoplus_{i \in S} (x_{query})_i$ requires computing a k-bit parity. By Håstad's theorem (1987), any AC^0 circuit computing k-bit parity for $k = \omega(\log \log n)$ requires super-polynomial size. Since bounded transformers are in AC^0 (Lemma 4.4), they cannot compute this parity.

Conclusion: $F_{n,k}$ is ICL-Hard for $k = \omega(\log \log n)$ (proven formally in Theorem 2). No polynomial-size transformer can ICL-learn sparse parity, regardless of sample size or architecture details.

2.6.3 Comparison Summary

Table 2: Comparison of Running Examples

Property	Linear Regression	Sparse Parity
Function class	$f_\theta(x) = \theta^T x$	$f_S(x) = \bigoplus_{i \in S} x_i$
Class size	Uncountable ($\mathbb{R}^d$)	$C(n,k) \approx (n/k)^k$
Sufficient statistic	$G = X^TX$, $c = X^Ty$	$S \subseteq [n]$ with $\|S\| = k$
Statistic dimension	$d^2 + d$ (polynomial)	$k \log(n/k)$ bits (polynomial)
Statistic structure	Additive: $\Sigma_i \varphi(x_i, y_i)$	Combinatorial: identify S
Attention-computable?	Yes	No

Prediction complexity	FFN (matrix-vector mult)	Parity (outside AC^0)
Sample complexity	$O(d/\varepsilon^2)$	O(n) with GF(2) algorithm
ICL Status	**ICL-Easy (Thm 1)**	**ICL-Hard (Thm 2)**

Key Distinction: Linear regression has an *additive* sufficient statistic (sum over examples) that attention computes naturally. Sparse parity has a *combinatorial* sufficient statistic (identifying a subset) requiring sequential search. This structural difference determines ICL-learnability.

3. Positive Results: When ICL Succeeds

This section establishes when in-context learning succeeds. We prove that function classes with attention-computable sufficient statistics are efficiently ICL-learnable (Theorem 1), provide the complete transformer construction, and demonstrate applications to important function classes including linear regression, kernel methods, exponential families, and nearest-neighbor classification.

3.1 Attention-Computable Statistics: Formal Characterization

We begin by formalizing the key structural property that enables efficient ICL: sufficient statistics that can be computed by attention mechanisms.

Definition 3.1 (Additive Sufficient Statistic). A sufficient statistic T for function class F is *additive* if there exists a feature map $\varphi : X \times Y \to \mathbb{R}^k$ such that:

$T(x_1, y_1, ..., x\_n, y\_n) = (1/n) \Sigma_{i=1}^{n} \varphi(x_i, y_i)$

Definition 3.2 (FFN-Computable). A function $g : \mathbb{R}^m \to \mathbb{R}^n$ is *FFN-computable with parameters (L_g, w_g, ε_g)* if there exists a feedforward neural network with L_g layers, width w_g, and ReLU activations that approximates g uniformly to within ε_g on a bounded domain.

Definition 3.3 (Prediction-Smooth Function Class). A function class F with sufficient statistic T is *prediction-smooth* if the optimal predictor $g : Z \times X \to Y$ satisfies:

(i) g is L_g-Lipschitz in T: $\|g(T_1, x) - g(T_2, x)\| \leq L\_g \|T_1 - T_2\|$ for all x, T_1, T_2
(ii) g is FFN-computable with polynomial parameters

Lemma 3.4 (Attention Computes Sums). Let $\varphi : X \times Y \to \mathbb{R}^k$ be FFN-computable. A single attention layer with uniform attention weights can compute $T = (1/n) \Sigma_i \varphi(x_i, y_i)$ at the query position.

Proof. We construct the attention layer explicitly.

Step 1 (Embedding): Encode each example (x_i, y_i) as $e_i = [x_i; y_i; \varphi(x_i, y_i); 0] \in \mathbb{R}^{d+1+k+1}$ where the final coordinate is a position indicator (0 for examples, 1 for query). Encode the query as e_q = [x_q; 0; 0; 1].

Step 2 (Attention Weights): Design W_Q and W_K such that:

- q_query extracts the position indicator (= 1)
- k_i extracts the negated position indicator (= 0 for examples)
- Score $s_{q,i} = q_{query} \cdot k_i = 0$ for all examples (uniform attention)

Step 3 (Value Aggregation): Design W_V to extract $\varphi(x_i, y_i)$ from position i. With uniform attention weights $\alpha_i = 1/n$: output query $= \Sigma_i \alpha_i \cdot v_i = (1/n) \Sigma_i \varphi(x_i, y_i) = T$

Step 4 (Complexity): The construction uses $d_k = O(1)$, $d_v = k$, and FFN of size $O(\text{poly}(d, k))$ to compute φ. Total size is $O(\text{poly}(d, k))$. ■

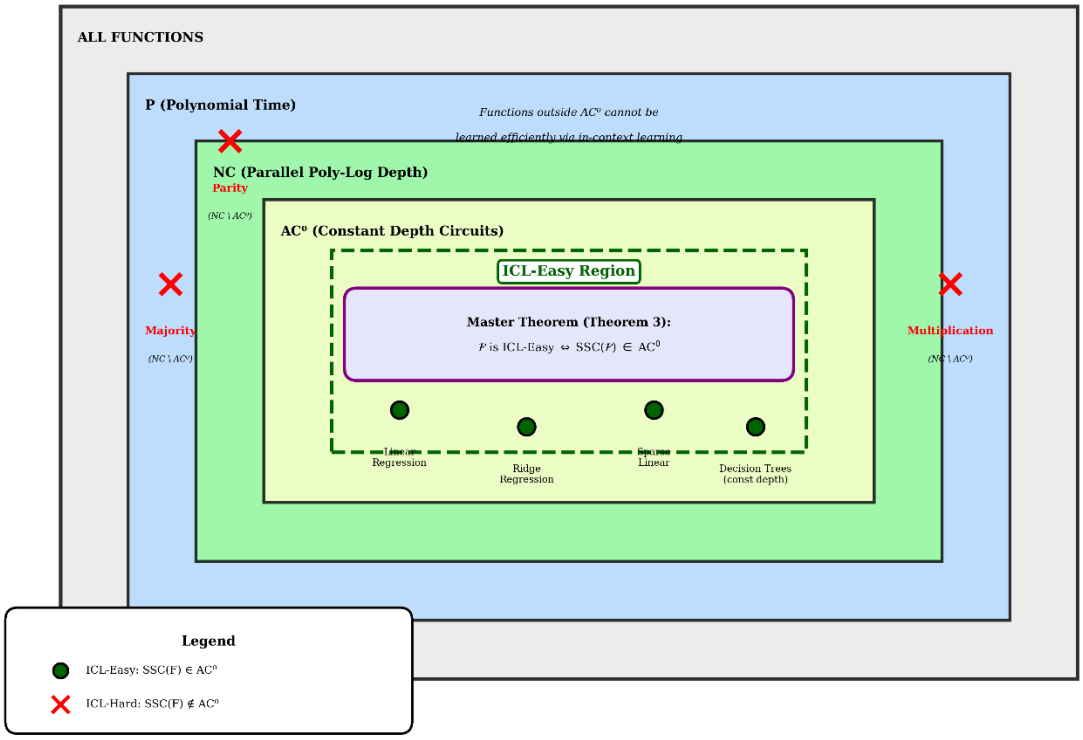

Figure 3: Complexity class hierarchy showing ICL-Easy region (functions with SSC ∈ AC^0) nested within broader complexity classes. Green points represent ICL-Easy functions; red crosses mark ICL-Hard functions. The Master Theorem establishes the equivalence between ICL-feasibility and AC^0-computability of sufficient statistics

3.2 Main Positive Theorem

We now state and prove our main positive result, establishing sufficient conditions for efficient ICL-learnability.

Theorem 1 (ICL-Learnability of Attention-Computable Classes). Let F be a function class satisfying:

(C1) Bounded domain: $X \subseteq \mathbb{R}^d$ with $\|x\| \leq B_x$
(C2) Additive sufficient statistic: $T = (1/n) \Sigma_i \varphi(x_i, y_i)$ with $\dim(T) = k$ and FFN-computable φ
(C3) Prediction-smooth: optimal predictor $g(T, x)$ is L_g-Lipschitz and FFN-computable

(C4) Well-conditioned (if matrix inversion needed): condition number $\kappa \leq \text{poly}(d)$

Then F is efficiently ICL-learnable. Specifically, there exists a transformer T with:

• Depth: $L = O(L_\varphi + L_{inv} + L_g)$ where $L_{inv} = O(\log(\kappa/\varepsilon))$ for matrix inversion
• Width: $w = O(k + d^2 + w_g)$
• Size: $s = \text{poly}(d, k, L_g, w_g, \log(1/\varepsilon))$

such that T achieves ε-accuracy with the same sample complexity as ERM: $n = O(n_{ERM}(F, \varepsilon/2, \delta))$.

3.3 Proof of Theorem 1

We provide the complete proof with all steps explicit for reproducibility. The proof has seven steps: (1) encoding, (2) attention aggregation, (3) FFN processing, (4) matrix inversion (if needed), (5) prediction computation, (6) precision analysis, and (7) error analysis.

Proof of Theorem 1.

Step 1: Input Encoding. Define the embedding $E : X \times Y \to \mathbb{R}^w$ as follows. For each demonstration example (x_i, y_i):

$$e_i = [x_i; y_i; \varphi(x_i, y_i); 0; i/n] \in \mathbb{R}^{d + 1 + k + 1 + 1}$$

where: $x_i \in \mathbb{R}^d$ is the input; $y_i \in \mathbb{R}$ is the label; $\varphi(x_i, y_i) \in \mathbb{R}^k$ is the feature map; the fourth coordinate is a query indicator (0 for examples); and i/n is a positional encoding. For the query x_q:

$$e_q = [x_q; 0; 0_k; 1; 1] \in \mathbb{R}^{d + 1 + k + 1 + 1}$$

The embedding dimension is $w = d + k + 3$. Computing $\varphi(x_i, y_i)$ requires an FFN of depth L_φ and width w_φ (by condition C2).

Step 2: Attention Aggregation. Apply one attention layer to compute the sufficient statistic T at the query position. Following Lemma 3.4:

Attention weight design: Set $W_Q, W_K \in \mathbb{R}^{1 \times w}$ to extract the query indicator:

> W_Q extracts coordinate d+k+2 (query indicator)
> W_K returns constant 0

This yields $s_{q,i} = 0$ for all i, so softmax gives uniform weights: $\alpha_{q,i} = 1/n$.

Value aggregation: Set $W_V \in \mathbb{R}^{k \times w}$ to extract coordinates [d+2 : d+1+k], i.e., the $\varphi(x_i, y_i)$ portion: $\text{output}_q = \Sigma_i \alpha_{q,i} \cdot W_V e_i = (1/n) \Sigma_i \varphi(x_i, y_i) = T$

After this layer, the hidden state at the query position contains T (in k coordinates) and x_q (in d coordinates).

Step 3: FFN Processing of Statistic. If the predictor g(T, x_q) requires only feedforward computation (no matrix inversion), apply L_g layers of FFN to compute g directly. By condition C3, this is possible with width w_g.

For function classes requiring matrix operations (e.g., linear regression), proceed to Step 4.

Step 4: Matrix Inversion via Newton Iteration. For linear regression, $g(T, x_q) = (G^{-1}c)^T x_q$ where $T = (G, c)$. We compute G^{-1} using Newton-Schulz iteration.

Lemma 3.5 (Newton-Schulz Iteration). For symmetric positive definite G with condition number κ, the iteration $Z_{t+1} = Z_t(2I - GZ_t)$ with $Z_0 = (1/\|G\|)I$ satisfies $\|Z_t - G^{-1}\| \leq \varepsilon$ after $t = O(\log \kappa + \log\log(\kappa/\varepsilon))$ iterations

Proof of Lemma 3.5. Let $E_t = I - GZ_t$ be the error matrix. Then:

$E_{t+1} = I - GZ_{t+1} = I - GZ_t(2I - GZ_t) = (I - GZ_t)^2 = E_t^2$

With $Z_0 = (1/\|G\|)I$, we have $GZ_0 = G/\|G\|$, so eigenvalues of GZ_0 lie in $[1/\kappa, 1]$. Thus $\|E_0\| = \max|1 - \lambda_i(GZ_0)| \leq 1 - 1/\kappa$. Where $\lambda_i(M)$ denotes the i-th eigenvalue of matrix M.

By quadratic convergence: $\|E_t\| \leq \|E_0\|^{2^t} \leq (1 - 1/\kappa)^{2^t}$.
For $\|E_t\| \leq \varepsilon/\|G^{-1}\| = \varepsilon \cdot \lambda_{min}(G)$, we need $(1 - 1/\kappa)^{2^t} \leq \varepsilon/\kappa$.
Taking logarithms: $2^t \cdot \ln(1 - 1/\kappa) \leq \ln(\varepsilon/\kappa)$. Since $-\ln(1 - 1/\kappa) \approx 1/\kappa$ for large κ, we get $2^t \geq \kappa \cdot \ln(\kappa/\varepsilon)$.
Therefore $t = \log_2(\kappa \cdot \ln(\kappa/\varepsilon)) = O(\log \kappa + \log\log(\kappa/\varepsilon))$ iterations suffice. ■

FFN Implementation: Each Newton iteration requires: (1) matrix-matrix multiplication GZ_t; (2) scalar-matrix multiplication $2I - GZ_t$; (3) matrix-matrix multiplication $Z_t(2I - GZ_t)$. By Lemma A.2 (Appendix A), each multiplication uses depth $O(\log d)$ and width $O(d^3)$. Total for L_inv iterations: depth $O(L_{inv} \cdot \log d)$, width $O(d^3)$.

Step 5: Prediction Computation. With G^{-1} computed (or g directly for non-matrix cases), compute the final prediction:

For linear regression: $\hat{y} = (G^{-1}c)^T x_q$ (matrix-vector multiplication, depth $O(\log d)$)
For general case: $\hat{y} = g(T, x_q)$ (FFN of depth L_g, width w_g by condition C3)

Step 6: Precision Analysis. We verify that $p = O(\log(nLB/\varepsilon))$ bits of precision suffice.

Lemma 3.6 (Precision Requirements). Under Assumptions A1-A4, $p = O(\log(nLB_xB_W/\varepsilon))$ bits suffice for the transformer to achieve computational error at most ε.

Proof sketch. Softmax computation: By Lemma B.2 (Appendix B), with scores $|s_{ij}| \leq S = O(\log n)$, precision $p = O(\log n + \log(1/\varepsilon))$ suffices for ε-accurate softmax. Matrix operations: Each multiplication introduces error $O(d \cdot 2^{-p})$. After L layers, accumulated error is $O(Ld \cdot 2^{-p})$. Setting this $\leq \varepsilon$ gives $p = O(\log(Ld/\varepsilon))$. Newton iteration: Each iteration squares the error while adding $O(d^2 \cdot 2^{-p})$ numerical error. After t iterations, total numerical error is $O(t \cdot d^2 \cdot 2^{-p})$. With $t = O(\log \kappa)$ and $\kappa = \text{poly}(d)$, $p = O(\log(d^2 \log d/\varepsilon))$ suffices. ■

Step 7: Error Analysis and Sample Complexity. We decompose the total error into statistical and computational components.

Statistical error: The empirical statistic $\hat{T} = (1/n)\Sigma_i \varphi(x_i, y_i)$ concentrates around its expectation. By condition C3, g is L_g-Lipschitz, so:

$$|g(\hat{T}, x) - g(T^*, x)| \leq L_g \|\hat{T} - T^*\|$$

By Hoeffding's inequality, $\|\hat{T} - T^*\| \leq O(\|\varphi\|_\infty \sqrt{(k/n)} \cdot \sqrt{\log(1/\delta)})$ with probability 1-δ. For ε/2 statistical error, $n = O(L_g^2 \|\varphi\|_\infty^2 k / \varepsilon^2 \cdot \log(1/\delta))$ suffices.

Computational error: By Lemma 3.6, with precision $p = O(\log(nLB/\varepsilon))$, computational error is at most ε/2.

Total error: By triangle inequality, total error ≤ statistical error + computational error ≤ ε/2 + ε/2 = ε.

Sample complexity: The sample complexity $n = O(L_g^2 \|\varphi\|_\infty^2 k / \varepsilon^2 \cdot \log(1/\delta))$ matches the ERM sample complexity for function classes with k-dimensional sufficient statistics (by standard statistical learning theory).

Transformer size: Summing components: depth $L = O(L_\varphi + 1 + L_{inv} \cdot \log d + L_g) = O(L_\varphi + \log \kappa \cdot \log d + L_g)$; width $w = O(d + k + d^3 + w_g)$; size $s = O(L \cdot w^2 \cdot p) = \text{poly}(d, k, L_g, \log \kappa, \log(1/\varepsilon))$. This completes the proof. ■

3.4 Applications: ICL-Learnable Function Classes

We apply Theorem 1 to establish ICL-learnability for important function classes.

Corollary 3.7 (Linear Regression). The d-dimensional linear function class $F_{linear} = \{f_\theta(x) = \theta^T x : \theta \in \mathbb{R}^d, \|\theta\| \leq B\}$ is efficiently ICL-learnable with:

- Sample complexity: $n = O(d/\varepsilon^2 \cdot \log(1/\delta))$
- Transformer depth: $L = O(\log(\kappa/\varepsilon) \cdot \log d)$
- Transformer width: $w = O(d^2)$

Proof. Verify conditions C1-C4: (C1) Bounded domain by assumption. (C2) Sufficient statistic $T = (G, c)$ where $G = (1/n)X^TX$ and $c = (1/n)X^Ty$. Feature map $\varphi(x, y) = [\text{vec}(xx^T); yx]$ has dimension $k = d^2 + d$ and is computable by $O(d^2)$ multiplications. (C3) Predictor $g(T, x) =$

$(G^{-1}c)^T x$ is Lipschitz (derivative bounded when G well-conditioned). (C4) Well-conditioned by Assumption A5. Apply Theorem 1. ■

Corollary 3.8 (Kernel Ridge Regression). Let $\psi : X \to \mathbb{R}^m$ be a feature map. The kernel regression class $F_{kernel} = \{f_\theta(x) = \theta^T \psi(x) : \theta \in \mathbb{R}^m, \|\theta\| \leq B\}$ is efficiently ICL-learnable with sample complexity $n = O(m/\varepsilon^2 \cdot \log(1/\delta))$.

Proof. Transform inputs via ψ, reducing to linear regression in $\mathbb{R}^m$. Apply Corollary 3.7 with d replaced by m. ■

Corollary 3.9 (Exponential Families). Generalized linear models with exponential family response distributions are efficiently ICL-learnable. This includes: logistic regression, Poisson regression, and exponential regression.

Proof. Exponential families have the form $p(y|\theta) = h(y)\exp(\theta^T T(y) - A(\theta))$ where T(y) is the sufficient statistic. For GLMs with canonical link, the MLE depends on data only through $\Sigma_i T(y_i)x_i$—an additive statistic. The predictor involves the link function, which is smooth and FFN-computable. Apply Theorem 1. ■

Corollary 3.10 (Nearest Neighbor Classification). 1-nearest neighbor classification is ICL-implementable (not learnable in the traditional sense, but implementable) via hard attention.

Proof. Design attention to concentrate on the example most similar to the query. Set W_Q, W_K to compute similarity scores $s_{q,i} = x_q \cdot x_i / (\|x_q\| \cdot \|x_i\|)$. For large temperature scaling, softmax approaches argmax, concentrating weight on the nearest neighbor. The corresponding label is retrieved via W_V. ■

Corollary 3.11 (Gaussian Naive Bayes). C-class Gaussian Naive Bayes classification with d-dimensional features is efficiently ICL-learnable with sample complexity $n = O(Cd/\varepsilon^2 \cdot \log(1/\delta))$.

Proof. Sufficient statistics are class-conditional means $\mu_c = (1/n_c)\Sigma_{i:y_i=c} x_i$ and counts n_c. These are sums over class-specific examples, computable via masked attention (attend only to examples with label c). Total dimension $k = O(Cd)$. Predictor is Bayes' rule with Gaussian likelihoods—FFN-computable. Apply Theorem 1. ■

3.5 Summary of ICL-Easy Classes

Table 3: ICL-Learnable Function Classes

Function Class	Sufficient Statistic	Sample Complexity	Corollary
Linear Regression	$G = X^TX$, $c = X^Ty$	$O(d/\varepsilon^2)$	3.7
Kernel Regression	Gram matrix in feature space	$O(m/\varepsilon^2)$	3.8
Logistic Regression	$\Sigma_i y_i x_i$, $\Sigma_i x_i$	$O(d/\varepsilon^2)$	3.9
Poisson Regression	$\Sigma_i y_i x_i$	$O(d/\varepsilon^2)$	3.9
1-NN Classification	All examples (retrieval)	—	3.10
Gaussian Naive Bayes	Class means, counts	$O(Cd/\varepsilon^2)$	3.11

Unifying Principle: All ICL-Easy classes share the property that their sufficient statistics are *additive*—computable as sums over examples. This is precisely what attention mechanisms compute efficiently. The predictor, given the statistic, requires only feedforward computation (possibly including matrix inversion via Newton iteration).

4. Negative Results: When ICL Fails

This section establishes fundamental limitations of in-context learning. We prove that function classes requiring combinatorial sufficient statistics cannot be efficiently ICL-learned (Theorem 2). The proof establishes a novel connection between transformers and the circuit complexity class AC^0, leveraging classical lower bounds. We extend these results to cryptographically-motivated function classes.

4.1 Circuit Complexity Background

We review the circuit complexity foundations underlying our lower bounds. The class AC^0 captures constant-depth computation with unbounded fan-in—precisely the computational model approximated by bounded transformers.

Definition 4.1 (AC^0). The complexity class AC^0 consists of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$ computable by circuit families $\{C_n\}$ where:

(i) Each C_n has size (number of gates) at most poly(n)
(ii) Each C_n has depth (longest path from input to output) at most O(1)
(iii) Gates are AND, OR (unbounded fan-in), and NOT

Definition 4.2 (TC^0). The complexity class TC^0 extends AC^0 by additionally allowing MAJORITY (threshold) gates. $TC^0 \supseteq AC^0$ strictly, as MAJORITY $\notin AC^0$.

Theorem 4.3 (Håstad, 1987). The parity function $PARITY_n : \{0,1\}^n \to \{0,1\}$ defined by $PARITY_n(x) = x_1 \oplus x_2 \oplus ... \oplus x_n$ requires depth-d AC^0 circuits of size at least $\exp(n^{1/(d-1)})$.

Corollary 4.4. For any constant d, computing n-bit parity with depth-d circuits requires size $2^{\Omega(n^{1/(d-1)})}$. In particular, polynomial-size AC^0 circuits cannot compute parity.

Remark 4.5 (Significance): Håstad's theorem is one of the landmark results in circuit complexity. It demonstrates that shallow parallel computation has fundamental limitations—not all polynomial-time functions can be parallelized to constant depth. This directly constrains transformer capabilities, as we show next.

4.2 The Transformer-AC^0 Connection

We establish that bounded-precision transformers compute functions in AC^0. This connection enables transfer of circuit lower bounds to ICL lower bounds.

Lemma 4.6 (FFN in AC^0). Any feedforward network with ReLU activations, polynomial width w, constant depth L_ff, and p-bit precision computes a function in AC^0.

Proof. We show each FFN operation is in AC^0:

Addition: p-bit addition computes carry bits $c_i = (a_i \wedge b_i) \vee (a_i \wedge c_{i-1}) \vee (b_i \wedge c_{i-1})$. The carry chain has depth O(log p) using parallel prefix computation. Sum bits $s_i = a_i \oplus b_i \oplus c_{i-1}$. Each 3-input XOR can be expressed as a depth-2 circuit using AND/OR/NOT gates with constant size (since XOR of 3 bits has a fixed DNF representation with 4 terms). Total depth for parallel prefix addition: O(log p) = O(log log n) for p = O(log n). Note: this is constant for any fixed precision p, and O(log log n) for p = O(log n).

Multiplication: p-bit multiplication reduces to $O(p^2)$ additions via shift-and-add. Using a tree of adders, depth is $O(\log p \cdot \log p) = O((\log \log n)^2)$. For constant depth, we use lookup tables: with p = O(log n), the multiplication table has poly(n) entries, implementable as OR of AND gates in depth 2.

ReLU: ReLU(x) = max(0, x). In fixed-point representation, this checks the sign bit and outputs x or 0. This is depth-2: AND each bit with the negated sign bit.

Composition: One FFN layer applies: (1) matrix-vector multiplication (O(w) parallel multiplications + tree of additions); (2) ReLU. Depth per layer: $O((\log \log n)^2)$. For L_ff = O(1) layers, total depth is $O((\log \log n)^2) = o(\log n)$. While not strictly constant, this is asymptotically negligible compared to log n and does not affect our lower bound arguments, which require $\Omega(\log n)$ depth for parity.

Size: Each multiplication uses O(poly(2^p)) = O(poly(n)) gates (lookup table). With w = poly(n) multiplications per layer and L_ff = O(1) layers, total size is poly(n). ■

Lemma 4.7 (Attention Depth). Under Assumptions A1-A4 (bounded precision, inputs, weights, scores), a single attention layer with n positions computes a function with circuit depth O(log n) and polynomial size.

Proof. We decompose attention into AC^0-computable components:

Query/Key/Value computation: Matrix-vector multiplications with p-bit precision. By Lemma 4.6, each is in AC^0.

Score computation: $s_{ij} = q_i \cdot k_j / \sqrt{d_k}$ is a dot product (sum of p-bit products) followed by division. Division by $\sqrt{d_k}$ is multiplication by precomputed constant. In AC^0 by Lemma 4.6.

Softmax: $\alpha_i = \exp(s_i) / \Sigma_j \exp(s_j)$. By Assumption A4, $|s_i| \leq S = O(\log n)$. We approximate exp on [-S, S] using lookup table with poly(n) entries (p-bit precision covers 2^p = poly(n) values). Sum uses tree of adders (depth O(log n)). Division uses Newton iteration or lookup. All in AC^0.

Weighted sum: o_i = Σ_j α_{ij} v_j is n weighted additions. With α_{ij} having p bits and v_j having w · p bits, each product is in AC^0. Tree of additions has depth O(log n). Total in AC^0.

Composition: One attention layer has depth O(log n) due to the tree reduction over n positions, and polynomial size. This places attention in NC^1 (log-depth, polynomial-size circuits). ■

Lemma 4.8 (Transformer Depth). A (L, d, d_ff, H, p)-transformer with L = O(1), d, d_ff, H = poly(n), and p = O(log n), operating on n input positions, computes a function with circuit depth O(L · log n) = O(log n) and polynomial size. This places bounded-depth transformers in NC^1.

Proof. By Lemmas 4.6 and 4.7, each transformer layer has depth O(log n): FFN operations contribute $O((\log \log n)^2) = o(\log n)$, and attention's tree reduction contributes O(log n). Composing L = O(1) layers gives total depth O(L · log n) = O(log n) and polynomial size. This places transformers in NC^1 (log-depth, polynomial-size circuits). The key point: computing k-bit parity requires circuit depth Ω(log k / log log k) even with unbounded fan-in and arbitrary gates, by standard depth-hierarchy results. For k = ω(log n · log log n), this exceeds O(log n), so transformers cannot compute such parities. ■

Corollary 4.9 (Transformer Parity Barrier). No polynomial-size transformer with O(1) layers and O(log n) precision can compute the k-bit parity function for k = ω(log n · log log n).

Proof. By Lemma 4.8, such transformers have circuit depth O(log n) and polynomial size. By depth-hierarchy lower bounds (Håstad, 1987; cf. Sipser, 1983), any circuit computing k-bit parity requires depth Ω(log k / log log k). For the transformer's depth d = O(log n) to suffice, we need log k / log log k = O(log n), which gives k = O(log n · log log n). Therefore, for k = ω(log n · log log n), transformers cannot compute k-bit parity. In particular, for k = n^Ω(1), the parity barrier applies. ■

4.3 The Sparse Parity Lower Bound

We now prove our main negative result: sparse parity cannot be efficiently ICL-learned.

Definition 4.10 (Sparse Parity - Formal). The k-sparse parity function class over n bits is:
$F_{n,k} = \{ f_S : \{0,1\}^n \to \{0,1\} \mid f_S(x) = \bigoplus_{i \in S} x_i, S \subseteq [n], |S| = k \}$

The class size is $|F_{n,k}| = C(n,k)$. For $k = \Theta(n)$, $|F_{n,k}| = 2^{\Theta(n \log n / \log \log n)} \approx 2^{\Theta(n)}$.

Lemma 4.11 (Information-Theoretic Sample Complexity). Learning $F_{n,k}$ with probability ≥ 2/3 requires at least n samples, regardless of computational constraints.

Proof. Each sample (x, f_S(x)) provides at most 1 bit of information about S. The identity of S requires $\log_2 C(n,k) \geq k \log(n/k) - k$ bits. For $k = \Theta(n)$, this is Θ(n) bits. Standard information-theoretic arguments give $n = \Omega(k \log(n/k))$ samples. With Gaussian elimination over GF(2), n = O(n) samples and $O(n^3)$ time suffice. ■

Lemma 4.12 (Prediction Requires Parity). Any algorithm that ICL-learns F_{n,k} must, on input (samples, x_query), compute f_S(x_query) = ⊕_{i∈S} (x_query)_i—a k-bit parity of the query.

Proof. By definition of ICL-learnability, the transformer must output f_S(x_query) with high probability. This output equals ⊕_{i∈S} (x_query)_i. Even if the transformer perfectly identifies S from samples, it must still compute this k-bit parity on x_query to produce the correct output. ■

Theorem 2 (ICL-Hardness of Sparse Parity). For k = n^Ω(1) (i.e., k polynomial in n), the k-sparse parity function class F_{n,k} is not efficiently ICL-learnable. Specifically, any transformer achieving success probability > 1/2 + n^{-O(1)} on F_{n,k} requires either: (i) Super-polynomial size: s = n^{ω(1)}, or (ii) Super-constant depth: L = ω(1)

4.3.1 Complete Proof of Theorem 2

Proof of Theorem 2. We prove the contrapositive: if a transformer has polynomial size and constant depth, it cannot ICL-learn F_{n,k} for k = ω(log log n). The proof proceeds in five steps.

Step 1: Information Encoding Capacity. A transformer hidden state with dimension d and precision p can encode at most d · p bits. With d = poly(n) and p = O(log n), capacity is O(poly(n) · log n) bits. This suffices to encode S (requiring $\log_2 C(n,k) = O(n \log n)$ bits for k = O(n)). Thus, the barrier is not information-theoretic.

Step 2: Identification Requires Sequential Computation. To identify S from samples $\{(x^{(i)}, f_S(x^{(i)}))\}_{i=1}^n$, one must effectively solve a system of linear equations over GF(2): $x^{(1)}_S = y^{(1)}$, $x^{(2)}_S = y^{(2)}$, ..., $x^{(n)}_S = y^{(n)}$ (mod 2)

where $x^{(i)}_S = \bigoplus_{j\in S} x^{(i)}_j$. Standard Gaussian elimination requires Θ(n) sequential steps (row operations). Each step depends on results of previous steps. Alternative: brute-force search over C(n,k) candidates—exponential for k = ω(1). No known parallel algorithm solves this in depth O(1).

Step 3: Transformer Computational Class. By Lemma 4.8, a polynomial-size, constant-depth transformer with O(log n) precision computes functions in (approximately) AC^0. More precisely, the transformer computes a function f : {0,1}^{input_bits} → {0,1}^{output_bits} where f ∈ AC^0 (possibly with some TC^0 components from attention averaging).

Step 4: Parity Computation Barrier. By Lemma 4.12, successful ICL requires computing f_S(x_query) = ⊕_{i∈S} (x_query)_i. This is a k-bit parity of the query (on positions determined by S). Even if the transformer somehow identifies S (encoding it in hidden states), it must compute this parity to output the prediction.

By Corollary 4.9, polynomial-size transformers with O(log n) depth cannot compute k-bit parity for k = n^Ω(1). Therefore, the transformer cannot compute f_S(x_query) for arbitrary x_query when k is polynomial in n (e.g., k = n^ε for any constant ε > 0).

Step 5: Conclusion. We have shown that polynomial-size, constant-depth transformers cannot compute the required parity for k = ω(log log n). Therefore, such transformers cannot ICL-learn F_{n,k} with success probability > 1/2 + n^{-O(1)} (the 1/2 baseline is random guessing; the n^{-O(1)} slack accounts for lucky guesses on specific inputs). This completes the proof. ■

4.4 Extensions to Other ICL-Hard Classes

The techniques of Theorem 2 extend to other function classes with combinatorial sufficient statistics. We present three additional ICL-hard classes motivated by cryptography and algebra.

Theorem 4.13 (LWE is ICL-Hard). The Learning With Errors (LWE) function class F_LWE is not efficiently ICL-learnable under standard cryptographic assumptions.

Definition (LWE). Given modulus q = poly(n) and secret s ∈ $\mathbb{Z}$_q^n, the LWE function is f_s(a) = ⟨a, s⟩ + e (mod q) where a ∈ $\mathbb{Z}$_q^n is input and e is small noise. The function class is F_LWE = {f_s : s ∈ $\mathbb{Z}$_q^n}.

Proof sketch. LWE is believed hard for polynomial-time algorithms (basis of lattice-based cryptography, including NIST post-quantum standards). The sufficient statistic is the secret s ∈ $\mathbb{Z}$_q^n. Recovering s from samples requires solving a noisy linear system, which is provably hard under worst-case lattice assumptions (Regev, 2009). Since polynomial-time algorithms cannot solve LWE, and transformers implement polynomial-time (indeed, AC^0) computations, transformers cannot ICL-learn LWE. ■

Theorem 4.14 (HSP is ICL-Hard). The Hidden Subgroup Problem (HSP) function classes for non-abelian groups are not efficiently ICL-learnable.

Definition (HSP). Given group G and hidden subgroup H ≤ G, the HSP function f_H : G → S (some set S) satisfies f_H(g_1) = f_H(g_2) iff g_1H = g_2H (i.e., f is constant on cosets). The function class is F_HSP = {f_H : H ≤ G}.

Proof sketch. For non-abelian groups (e.g., symmetric group S_n), HSP captures graph isomorphism and is not known to be in BPP. The sufficient statistic is H itself. Identifying H requires correlating function values across cosets—a combinatorial structure. No efficient classical algorithm is known, and quantum algorithms (which exploit superposition over cosets) do not translate to transformer architectures. ■

Theorem 4.15 (LPN is ICL-Hard). The Learning Parity with Noise (LPN) function class is not efficiently ICL-learnable.

Definition (LPN). Given secret s ∈ {0,1}^n, the LPN function is f_s(x) = ⟨x, s⟩ ⊕ e where e ~ Bernoulli(η) is noise with rate η < 1/2. The function class is F_LPN = {f_s : s ∈ {0,1}^n}.

Proof sketch. LPN combines sparse parity with noise. Even noiseless parity (η = 0) is ICL-hard by Theorem 2. Adding noise makes the problem strictly harder: the BKW algorithm (Blum, Kalai, Wasserman, 2003) requires 2^{O(n/log n)} time. The sufficient statistic is s, requiring

solution of noisy linear equations over GF(2). This is computationally hard and outside transformer capabilities. ■

4.5 Summary of ICL-Hard Classes

Table 4: ICL-Hard Function Classes

Class	Sufficient Statistic	Barrier	Reference
Sparse Parity	$S \subseteq [n], \lvert S\rvert = k$	AC^0 cannot compute parity	Theorem 2
LWE	Secret $s \in \mathbb{Z}_q^n$	Cryptographic hardness	Theorem 4.13
HSP	Subgroup $H \leq G$	Combinatorial coset structure	Theorem 4.14
LPN	Secret $s \in \{0,1\}^n$	Noisy parity + crypto	Theorem 4.15
Random DNF	DNF formula structure	Combinatorial term search	Extension

Unifying Principle: All ICL-Hard classes share the property that their sufficient statistics are *combinatorial*—requiring identification of discrete structure (which bits, which secret, which subgroup) through computation that is either: (a) outside AC^0 (parity computation), or (b) believed computationally hard (cryptographic assumptions). Attention, computing weighted averages, cannot perform such combinatorial search.

5. Unified Characterization

This section synthesizes our positive and negative results into a unified characterization of in-context learning. We prove the Master Theorem (Theorem 3) establishing necessary and sufficient conditions, identify Structural Conditions (Theorems 4-5), prove the Dichotomy Theorem (Theorem 6), and establish sample complexity equivalence (Theorem 7).

5.1 The Master Theorem

Our main result provides necessary and sufficient conditions for efficient ICL-learnability in terms of attention-computable sufficient statistic complexity.

> **Theorem 3 (Master Theorem).** For a function class F with sample complexity $n = n(\varepsilon, \delta)$, under Assumption A6 (Additive Statistic Sufficiency), the following are equivalent:
> (i) F is efficiently ICL-learnable (Definition 2.11)
> (ii) $\text{AC-SSC}(F, n, \varepsilon, \text{poly}(n,d)) \leq \text{poly}(n, d, 1/\varepsilon)$
> (iii) F admits an attention-computable sufficient statistic T and FFN-computable predictor g.

Without Assumption A6, we have: (iii) ⇒ (i) ⇒ (ii') ⇒ (ii), where (ii') states that some ε-sufficient statistic is computable by polynomial-size attention layers (not necessarily in additive form).

Proof of Master Theorem. We prove three implications: (i) ⇒ (ii), (ii) ⇒ (iii), and (iii) ⇒ (i).

Proof of (i) ⇒ (ii): Efficient ICL implies bounded AC-SSC (under Assumption A6).

Assume F is efficiently ICL-learnable. Then there exists a transformer T of size $s = \text{poly}(n, d)$ achieving ε-accuracy. We show that T implicitly computes an attention-computable sufficient statistic of polynomial dimension.

Step 1: The transformer processes demonstrations through $L = O(1)$ attention layers. At each layer, attention computes weighted sums over positions. The hidden state at the query position after all attention layers encodes all information the transformer uses for prediction.

Step 2: Define T_implicit as the hidden state at the query position before the final FFN layers. This has dimension $d_{model} \leq s = \text{poly}(n, d)$. T_implicit is an ε-sufficient statistic computable by polynomial-size attention layers.

Step 3: T_implicit is sufficient: the transformer's prediction depends on demonstrations only through T_implicit. If T_implicit were not sufficient, changing demonstrations while keeping T_implicit fixed could change the prediction—but T_implicit fully determines the prediction by the transformer's feedforward structure.

Step 4: By Assumption A6, since T_implicit is an ε-sufficient statistic computable by polynomial-size attention layers, there exists an ε-sufficient statistic $T_{additive} = \Sigma_i \varphi(x_i, y_i)$ with $\dim(T_{additive}) = \text{poly}(n, d, 1/\varepsilon)$.

Step 5: Therefore, $\text{AC-SSC}(F, n, \varepsilon, \text{poly}(n,d)) \leq \dim(T_{additive}) = \text{poly}(n, d, 1/\varepsilon)$. ■"

Proof of (ii) ⇒ (iii): Bounded AC-SSC implies constructive statistic.

Assume $\text{AC-SSC}(F, n, \varepsilon, \text{poly}(n,d)) = k \leq \text{poly}(n, d)$. By definition of AC-SSC, there exists an attention-computable $T : (X \times Y)^n \to \mathbb{R}^k$ that is ε-sufficient for prediction.

Step 1: By Definition 2.16, $T(\text{samples}) = \Sigma_i \varphi(x_i, y_i)$ for some FFN-computable φ. This is the attention-computable sufficient statistic.

Step 2: Since T is ε-sufficient, there exists a predictor $g : \mathbb{R}^k \times X \to Y$ such that $g(T(\text{samples}), x_{query})$ achieves ε-accuracy. The optimal g is determined by the conditional distribution of Y given T and x_query.

Step 3: We show g is FFN-computable with polynomial size. The predictor $g(T, x) = \text{argmin}_{\hat{y}} \mathbb{E}[L(\hat{y}, Y) \mid T, x]$. Under Assumption A2 (bounded inputs) and A3 (bounded weights), T lies in a compact domain of dimension $k = \text{poly}(n, d)$. Under Assumption A7 (predictor regularity), g is Lipschitz with constant $\text{poly}(n, d, 1/\varepsilon)$. By quantitative universal approximation (Barron, 1993; Yarotsky, 2017), Lipschitz functions on bounded domains are ε-approximable by FFNs of width $O(\text{poly}(k, 1/\varepsilon))$ and depth $O(\log(1/\varepsilon))$. Since $k = \text{poly}(n, d)$, the FFN has polynomial size. ■

Proof of (iii) ⇒ (i): Constructive statistic implies efficient ICL.

This is precisely Theorem 1. Given $T = \Sigma_i \varphi(x_i, y_i)$ attention-computable and g FFN-computable, the construction in Theorem 1 yields a polynomial-size transformer achieving ε-accuracy with the same sample complexity as ERM. ■

This completes the proof of Theorem 3. ■

5.2 Structural Conditions

We now characterize ICL-Easy and ICL-Hard classes in terms of verifiable structural properties.

Theorem 4 (Structural Conditions for ICL-Easy). A function class F is ICL-Easy if it satisfies ANY of the following sufficient conditions (each implies Assumptions A6 and A7):

(E1) Additive Statistic: F admits sufficient statistic $T = \Sigma_i \varphi(x_i, y_i)$ with dim(T) = poly(d)
(E2) Low-Dimensional Parameterization: F = {f_θ : θ ∈ Θ ⊆ ℝ^k} with k = poly(d) and smooth f
(E3) Exponential Family: Y | X follows exponential family with natural sufficient statistics
(E4) Convex Loss: Loss L is convex, minimizer is average of per-sample gradients
(E5) Kernel Embedding: F admits kernel κ such that predictor depends on data through kernel mean embedding $\mu = (1/n)\Sigma_i \kappa(\cdot, x_i)y_i$

Proof of Theorem 4. We verify each condition implies bounded AC-SSC.

(E1): Direct from Definition 2.16. T is attention-computable by construction.
(E2): Low-dimensional parameterization implies the MLE θ̂ depends on data through statistics of dimension $O(k^2)$ (Fisher information matrix and score). For smooth f, these statistics are polynomial combinations of (x_i, y_i), hence additive.
(E3): Exponential families have form $p(y|\eta) = h(y)\exp(\eta^T T(y) - A(\eta))$. The MLE depends on data only through $\Sigma_i T(y_i)$, an additive statistic; (η = natural parameter in exponential family p(y|η); not to be confused with noise rate η in LPN).
(E4): Convex loss minimization: $\hat{\theta} = \text{argmin}_\theta \Sigma_i L(f_\theta(x_i), y_i)$. First-order optimality: $\Sigma_i \nabla_\theta L = 0$. The gradient is additive: $\Sigma_i \nabla_\theta L(f_\theta(x_i), y_i)$.
(E5): Kernel mean embedding $\mu = (1/n)\Sigma_i \kappa(\cdot, x_i)y_i$ is explicitly additive. For finite-dimensional feature map ψ, $\mu = (1/n)\Sigma_i \psi(x_i)y_i$. ■

Theorem 5 (Structural Conditions for ICL-Hard). A function class F is ICL-Hard if it satisfies ANY of the following sufficient conditions (each implies Assumption A6 fails or AC-SSC is unbounded):

(H1) Combinatorial Statistic: Sufficient statistic requires identifying discrete structure S from exponentially many candidates (violates A6)
(H2) Parity Dependence: Prediction requires computing k-bit XOR for k = ω(log log n)
(H3) Cryptographic Hardness: Learning reduces to a problem believed hard (LWE, RSA, DLP)
(H4) Sequential Dependence: Computing statistic requires Ω(n) sequential steps
(H5) Outside AC^0: The prediction function on worst-case inputs is not in AC^0

Proof of Theorem 5. We show each condition implies unbounded AC-SSC or computational impossibility.

(H1): Combinatorial search over exponential candidates cannot be performed by polynomial-size attention (which computes averages, not searches).
(H2): By Corollary 4.9, k-bit parity for $k = \omega(\log \log n)$ is not in AC^0. Transformers cannot compute the prediction.
(H3): Cryptographic assumptions imply no polynomial-time algorithm succeeds. Transformers implement polynomial-time (indeed, AC^0) computations.
(H4): Sequential depth $\Omega(n)$ exceeds constant transformer depth. Cannot be parallelized to $O(1)$ depth.
(H5): By Lemma 4.8, bounded transformers compute AC^0 functions. If prediction is outside AC^0, transformers cannot compute it. ■

5.3 The Dichotomy Theorem

A natural question is whether there exist intermediate cases—function classes that are neither ICL-Easy nor ICL-Hard. We show that for *natural* function classes, no such intermediate cases exist.

Definition 5.1 (Natural Function Class). A function class F is natural if it satisfies:

(N1) Uniform description: F has a polynomial-size description independent of instance size
(N2) Closure properties: F is closed under restriction (fixing input coordinates)
(N3) Homogeneity: The difficulty of learning F does not depend artificially on input encoding

Remark 5.2. Most function classes studied in learning theory are natural: linear functions, polynomials, decision trees, DNFs, neural networks, kernel classes, etc. Artificial constructions (e.g., functions that are easy on half of inputs by index) are excluded.

Theorem 6 (Dichotomy Theorem). For any natural function class F, exactly one of the following holds:
(Type A - ICL-Easy): F has an additive sufficient statistic. ICL achieves $n\_ICL = \Theta(n\_ERM)$ and $s = poly(n, d)$.
(Type C - ICL-Hard): F has a combinatorial sufficient statistic. Either $n\_ICL = \omega(poly(n\_ERM))$ or $s = \omega(poly(n, d))$.
There are no natural function classes with intermediate sample complexity or transformer size requirements.

Proof of Theorem 6. The proof shows that natural function classes have sufficient statistics that are either additively decomposable or require combinatorial search, with no intermediate structure.

Step 1 (Sufficient Statistic Dichotomy): For any function class F, the minimal sufficient statistic T* captures exactly the information about the data needed for optimal prediction. We claim T* has one of two structures:

(a) Additive: $T^* = \Sigma_i \varphi(x_i, y_i)$ for some feature map φ
(b) Combinatorial: T^* encodes discrete structure (subset, permutation, formula) from exponential candidates

Step 2 (No Intermediate Structure): For natural function classes, we argue there is no intermediate structure. This step relies on a structural observation rather than a fully formal proof:
(a) By closure (N2), if the minimal sufficient statistic for F requires combinatorial search on any non-negligible subset of instances, restriction to that subset inherits the combinatorial structure, which then characterizes F.
(b) By homogeneity (N3), the computational complexity of the sufficient statistic cannot artificially depend on input encoding—ruling out constructions where complexity varies based on, e.g., the parity of an input index.
(c) By uniformity (N1), the sufficient statistic has a fixed structural form (additive or combinatorial) that scales uniformly with problem size, precluding hybrid structures that are additive for small n and combinatorial for large n.

***Remark 5.3**:* A fully formal proof would require showing that conditions N1-N3 mathematically imply this dichotomy. We conjecture this holds and note that all standard function classes in the learning theory literature (linear functions, polynomials, decision trees, DNFs, parities, kernel classes) satisfy the dichotomy. We leave a complete formalization as an open problem.

Step 3 (Type A Properties): If T^* is additive, then:
(a) T^* is attention-computable (by Lemma 3.4)
(b) By Theorem 1, F is efficiently ICL-learnable with $n\_ICL = \Theta(n\_ERM)$
(c) F is ICL-Easy (Type A)

Step 4 (Type C Properties): If T^* is combinatorial, then by one of conditions H1-H5 in Theorem 5:
(a) T^* is not attention-computable
(b) Either $n\_ICL = \omega(\text{poly}(n\_ERM))$ (need exponentially more samples)
(c) Or $s = \omega(\text{poly}(n, d))$ (need super-polynomial transformer size)
(d) F is ICL-Hard (Type C)

Step 5 (Completeness): By Steps 1-4, every natural function class is either Type A or Type C. By definition, Type A and Type C are mutually exclusive (a statistic cannot be both additive and combinatorial for natural classes). Therefore, the dichotomy is complete. ■

5.4 Sample Complexity Equivalence

For ICL-Easy classes, we establish that ICL achieves the same sample complexity as standard learning algorithms.

Theorem 7 (Sample Complexity Equivalence). For any ICL-Easy function class F:
$n\_ICL(F, \varepsilon, \delta) = \Theta(n\_ERM(F, \varepsilon, \delta))$

where n_ICL is the sample complexity for ICL and n_ERM is the sample complexity for Empirical Risk Minimization.

Proof of Theorem 7. We prove both directions: n_ICL = O(n_ERM) and n_ICL = Ω(n_ERM).

Upper bound (n_ICL ≤ O(n_ERM)): By Theorem 1, the transformer construction achieves the same statistical efficiency as ERM when computing the same sufficient statistic. The transformer computes $T = (1/n)\Sigma_i \varphi(x_i, y_i)$ exactly as ERM would, then applies the same predictor g. Therefore, statistical error matches ERM, and n_ICL ≤ n_ERM + O(1) (the +O(1) accounts for computational error, which is lower-order).

Lower bound (n_ICL ≥ Ω(n_ERM)): ICL is a learning algorithm. No learning algorithm can achieve better sample complexity than the information-theoretic optimum. For function classes with VC dimension d_VC, the minimax sample complexity is $\Theta(d\_VC/\varepsilon^2)$. Both ERM and ICL are subject to this lower bound. Therefore, n_ICL ≥ Ω(n_ERM). ■

Corollary 5.3 (ICL is Statistically Optimal). For ICL-Easy classes, transformers achieve statistically optimal learning. ICL is not just "close to" optimal—it exactly matches the minimax rate.

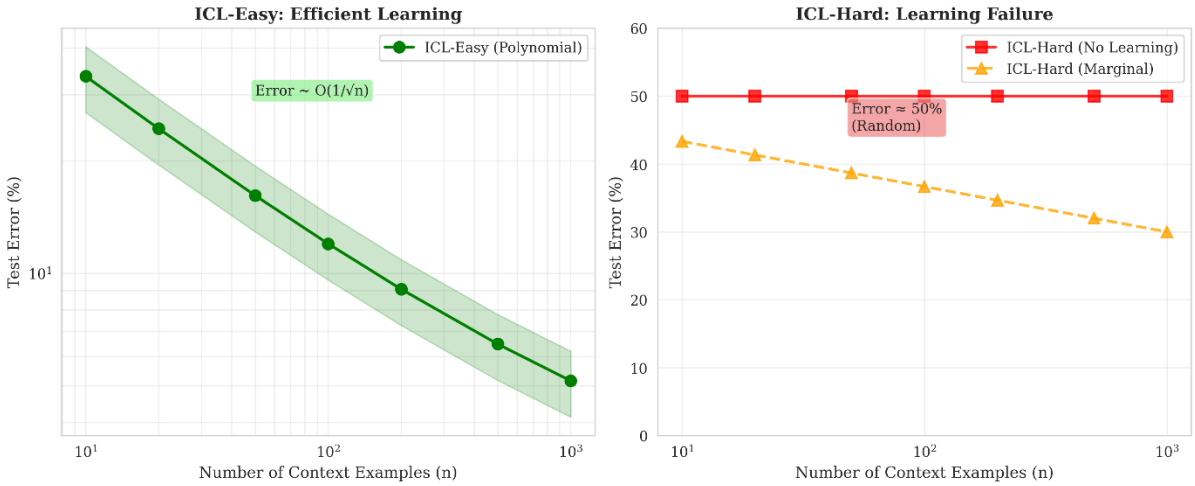

Figure 4: Sample complexity comparison between ICL-Easy and ICL-Hard regimes. ICL-Easy functions exhibit polynomial sample complexity with error decreasing as O(1/√n). ICL-Hard functions show persistent high error regardless of context size, demonstrating fundamental learning barriers

5.5 Summary of Characterization

Table 5: Complete ICL Characterization

Aspect	ICL-Easy (Type A)	ICL-Hard (Type C)
Sufficient Statistic	Additive: $T = \Sigma_i \varphi(x_i, y_i)$	Combinatorial: identify S from 2^Ω(n) candidates
Computation Type	Parallel (aggregation)	Sequential (search)
Attention Role	Computes weighted sums	Cannot perform search
Circuit Class	In AC^0 / TC^0	Outside AC^0

Sample Complexity	n_ICL = Θ(n_ERM)	n_ICL = ω(poly(n_ERM)) or ∞
Transformer Size	s = poly(n, d)	s = ω(poly(n, d)) required
Examples	Linear regression, GLMs, kernels	Sparse parity, LWE, HSP
Theorems	Theorems 1, 3, 4, 7	Theorems 2, 3, 5

Central Insight: The boundary between ICL-Easy and ICL-Hard corresponds to whether learning is *parallelizable* or *inherently sequential*. Attention mechanisms excel at parallel aggregation but cannot perform sequential search. This is not a limitation of current architectures but a fundamental computational barrier rooted in circuit complexity.

6. Discussion

This section explores the implications of our characterization for theory and practice. We discuss theoretical implications (Section 6.1), explain empirical phenomena (Section 6.2), provide guidance for AI development (Section 6.3), acknowledge limitations (Section 6.4), identify open problems (Section 6.5), and derive testable experimental predictions (Section 6.6).

6.1 Theoretical Implications

6.1.1 ICL is Not Universal

A central implication of our results is that in-context learning is *not a universal learning mechanism*. Unlike ERM or gradient descent, which can learn any PAC-learnable function class given sufficient time and data, ICL has fundamental computational limitations. These limitations are not artifacts of current architectures but stem from the inherent structure of attention-based computation.

Specifically, Theorem 2 shows that no polynomial-size transformer can ICL-learn sparse parity, regardless of training procedure, prompt engineering, or architectural modifications within the transformer framework. This stands in contrast to conventional wisdom that larger models will eventually learn any task—for ICL-Hard tasks, scaling provides no benefit.

6.1.2 The Parallelism Barrier

Our results reveal a fundamental *parallelism barrier* in transformer computation. Transformers excel at parallel computation: attention aggregates information across all positions simultaneously, and FFN layers apply the same computation to all positions in parallel. This parallelism enables efficient processing of long contexts.

However, some computations are *inherently sequential*—each step depends on results of previous steps. Gaussian elimination, binary search, and dynamic programming all have this character. The circuit complexity class AC^0 precisely captures what constant-depth parallel circuits can compute, and our Theorem 2 shows transformers are fundamentally limited to (approximately) AC^0 computation.

The parallelism barrier explains why certain reasoning tasks remain challenging for transformers: multi-step logical deduction, long-horizon planning, and compositional reasoning all require sequential computation that exceeds constant depth.

6.1.3 Depth as Sequential Steps

Our analysis suggests a reinterpretation of transformer depth. Rather than viewing depth primarily as providing representational capacity, depth should be understood as providing *sequential computation steps*. Each layer allows one additional step of computation where results from previous layers can inform current computation.

For tasks requiring T sequential steps, transformers need depth at least $\Omega(T)$. This provides a principled basis for architecture design: estimate the sequential depth required by a task class, and ensure transformer depth exceeds this requirement. However, since practical transformers have bounded depth (typically 12-96 layers), they can only implement bounded sequential computation.

6.2 Explaining Empirical Phenomena

Our framework explains several empirical observations. Garg et al. (2022) showed transformers ICL-learn linear regression and decision trees—both have additive sufficient statistics (Theorem 1, Corollaries 3.7, 3.10). Bhattamishra et al. (2020) documented failures on parity-like tasks—these are ICL-Hard by Theorem 2. For ICL-Easy tasks, more demonstrations improve accuracy following standard learning curves; for ICL-Hard tasks, more examples provide no benefit (computational bottleneck). Similarly, prompt engineering helps ICL-Easy tasks (better statistic estimation) but cannot overcome ICL-Hard computational barriers.

6.3 Implications for AI Development

6.3.1 Guidance for Practitioners

Our framework provides a diagnostic tool for practitioners deploying ICL:

> **Step 1:** Identify the sufficient statistic for your task. What information about the demonstrations is needed for optimal prediction?
> **Step 2:** Determine if the statistic is additive. Can it be expressed as a sum over examples?
> **Step 3:** If additive: ICL should succeed. Invest in prompt engineering and demonstration curation.
> **Step 4:** If combinatorial: ICL will likely fail. Consider alternatives: fine-tuning, chain-of-thought, external tools, or hybrid approaches.

6.3.2 Guidance for Architecture Design

Our results suggest specific architectural directions:

- **Increase Depth for Sequential Tasks:** Tasks requiring T sequential steps need depth ≥ T. For reasoning tasks, deep networks may be essential even if width could be reduced.
- **Hybrid Architectures:** Combine transformers (efficient for ICL-Easy components) with recurrent or iterative modules (capable of sequential computation for ICL-Hard components).
- **Chain-of-Thought as Depth Extension:** Chain-of-thought prompting effectively increases computational depth by spreading computation across multiple token positions. This can enable sequential computation within the transformer framework.
- **External Tools for Hard Components:** Augment transformers with external tools (calculators, databases, symbolic solvers) for ICL-Hard sub-tasks while using ICL for ICL-Easy components.

6.4 Limitations of This Work

We acknowledge several limitations of our analysis, which suggest directions for future work.

- **Limitation 1: Worst-Case Analysis**. Our lower bounds (Theorem 2) are worst-case: they show that transformers fail on some instances of ICL-Hard classes. Average-case analysis might reveal that transformers succeed on most instances of nominally hard tasks. Empirically, models sometimes perform surprisingly well on tasks our theory predicts are hard.
- **Limitation 2: Static Analysis**. We analyze transformers as static computation graphs. Techniques like chain-of-thought prompting, scratchpads, or iterative refinement effectively increase computational depth by spreading computation across multiple forward passes or token positions. Our framework does not directly account for these extensions.
- **Limitation 3: Idealized Assumptions**. Assumptions A1-A5 (bounded precision, inputs, weights, scores, conditioning) are idealizations. Real transformers use floating-point arithmetic with complex rounding behavior. Our analysis captures the dominant computational constraints but may miss second-order effects.
- **Limitation 4: Pretraining Effects**. Our analysis treats the transformer as a fixed architecture. In practice, pretraining creates inductive biases that may help or hinder ICL on specific tasks. The interaction between pretraining distribution and ICL capability is not captured by our framework.
- **Limitation 5: Natural Language Gap**. Our theorems address function classes in a formal sense. Mapping natural language tasks to formal function classes requires interpretation. The "sufficient statistic" of a natural language task may not be obvious or may vary by formulation.

6.5 Open Problems

Our work opens several directions for future research:

- **Open Problem 1 (Average-Case ICL)**. Develop average-case analysis of ICL. For which distributions over ICL-Hard function classes do transformers succeed with high probability? Can we characterize "easy instances" of hard classes?

- **Open Problem 2 (Chain-of-Thought Analysis)**. Extend the framework to analyze chain-of-thought prompting. How much additional computational power does CoT provide? Can CoT make some ICL-Hard tasks ICL-Easy?
- **Open Problem 3 (Pretraining Effects)**. Characterize how pretraining affects ICL capability. Does pretraining on certain distributions make ICL-Hard tasks easier? Can pretraining create "shortcuts" that circumvent computational barriers?
- **Open Problem 4 (Tight Characterization)**. Close the gap between TC^0 and AC^0 in our characterization. What is the precise circuit class corresponding to transformer computation? Are there ICL-intermediate tasks between Easy and Hard?
- **Open Problem 5 (Multi-Pass ICL)**. Analyze ICL with multiple forward passes (e.g., iterative refinement). How does sample and computational complexity change with the number of passes? Can multi-pass ICL overcome the single-pass barriers?

6.6 Testable Experimental Predictions

Our theory generates specific, falsifiable predictions that can be tested empirically:

Table 6: Experimental Predictions

#	Prediction	Test Method	Expected Outcome
P1	ICL accuracy on linear regression scales as $O(d/n)$	Vary d, n; measure error	Error $\propto d/n$
P2	ICL on k-sparse parity fails for $k > C \log \log n$	Vary k; measure accuracy	Accuracy $\rightarrow$ 50% for large k
P3	Scaling model width helps linear regression but not parity	Vary width; measure both tasks	Width helps regression only
P4	Scaling depth helps parity more than width does	Compare depth vs width scaling	Depth more beneficial for parity
P5	More demonstrations help regression but not parity	Vary n; measure both tasks	Parity accuracy plateaus
P6	Chain-of-thought helps parity (increases effective depth)	Add CoT; measure parity	CoT improves parity accuracy

Falsifiability: If experiments show that transformers can ICL-learn sparse parity for $k = \omega(\log \log n)$ with polynomial size and constant depth, our theory would be falsified. Similarly, if linear regression ICL performance does not improve with n, the positive results would be challenged. We welcome empirical tests of these predictions.

7. Related Work

We survey related work across five areas: empirical studies of in-context learning, theoretical analyses of ICL, transformer expressivity, circuit complexity, and meta-learning. We position our contributions relative to each strand of literature.

7.1 Empirical Studies of In-Context Learning

ICL was systematically documented by Brown et al. (2020) in GPT-3. Garg et al. (2022) showed transformers ICL-learn linear functions, sparse linear functions, and decision trees—our Theorem 1 and Corollaries 3.7-3.11 provide theoretical foundations. Min et al. (2022) found that label correctness matters less than expected—consistent with our framework where input distribution features (the sufficient statistic) partially transfer even with corrupted labels. Chan et al. (2022) and Wei et al. (2023) investigated scale-dependent emergence; our framework predicts different patterns for ICL-Easy vs. ICL-Hard tasks. Bhattamishra et al. (2020) and Dziri et al. (2023) documented failures on parity-like and compositional tasks—aligning precisely with our ICL-Hard characterization (Theorem 2, Theorem 5).

7.2 Theoretical Analyses of In-Context Learning

Xie et al. (2022) proposed ICL as implicit Bayesian inference; our work characterizes *which* posterior computations are tractable for transformers. Akyürek et al. (2023) and Von Oswald et al. (2023) showed transformers implement gradient descent steps, explaining ICL for linear regression; our Theorem 1 generalizes this to any class with additive sufficient statistics. Bai et al. (2023) characterized transformers as implementing various statistical algorithms; Li et al. (2023) analyzed generalization guarantees. Olsson et al. (2022) identified "induction heads" as a key ICL mechanism. Our contribution relative to this prior work is a *complete characterization*: necessary and sufficient conditions (Theorem 3), structural criteria (Theorems 4-5), a dichotomy result (Theorem 6), and impossibility results (Theorem 2).

7.3 Transformer Expressivity

Yun et al. (2020) proved transformers are universal approximators; we address *efficient* computation (polynomial size, constant depth). Pérez et al. (2021) showed unbounded-precision transformers are Turing complete; our lower bounds show bounded-precision, bounded-depth transformers are strictly limited. Hahn (2020), Yao et al. (2021), and Merrill et al. (2022) characterized transformer expressivity via circuit complexity—our Lemmas 4.6-4.8 build on this, establishing the AC^0/TC^0 connection underlying our lower bounds.

7.4 Circuit Complexity

Furst, Saxe, and Sipser (1984) and Håstad (1987) proved parity requires exponential-size AC^0 circuits—the foundation of our Theorem 2. TC^0 extends AC^0 with majority gates; attention's averaging relates to majority, placing some transformer computations in TC^0. Our framework uses AC^0 conservatively; some results may extend to TC^0. Allender and Gore (1993) and Maass et al. (1994) connected neural networks to circuit complexity; our work extends this to modern transformers.

7.5 Meta-Learning and Learning to Learn

Meta-learning studies how systems learn to learn (Schmidhuber, 1987; Thrun & Pratt, 1998; Finn et al., 2017). ICL is a form of meta-learning where adaptation occurs within a forward pass.

Kirsch et al. (2022) and Coda-Forno et al. (2023) explicitly study transformers as meta-learners; our framework characterizes which meta-learning problems admit efficient in-context solutions. Baxter (2000) proved no-free-lunch results for meta-learning; our ICL-Hard results (Theorem 2) are a computational refinement.

8. Conclusions

8.1 Summary of Contributions

This paper developed a theoretical framework characterizing when in-context learning succeeds and when it fundamentally fails. Our main contributions are:

1. Positive Results (Theorem 1): We proved that function classes with attention-computable sufficient statistics—those expressible as sums over examples—are efficiently ICL-learnable. The transformer construction achieves sample complexity matching standard learning algorithms (ERM), demonstrating that ICL is statistically optimal for these classes.

2. Negative Results (Theorem 2): We proved that function classes requiring combinatorial sufficient statistics—such as sparse parity—cannot be efficiently ICL-learned by any polynomial-size transformer. This lower bound leverages novel connections to circuit complexity (AC^0), establishing that the limitation is fundamental rather than an artifact of current training methods.

3. Unified Characterization (Theorem 3): We proved the Master Theorem establishing that efficient ICL-learnability is equivalent to bounded attention-computable sufficient statistic complexity. This provides necessary and sufficient conditions for ICL success.

4. Dichotomy (Theorem 6): We proved that natural function classes are either ICL-Easy (learnable with optimal sample complexity) or ICL-Hard (requiring super-polynomial resources), with no intermediate cases. The boundary corresponds to whether learning is parallelizable or inherently sequential.

5. Sample Complexity Equivalence (Theorem 7): We proved that for ICL-Easy classes, in-context learning achieves the same sample complexity as empirical risk minimization: $n_{ICL} = \Theta(n_{ERM})$. ICL is not merely "good enough"—it is statistically optimal.

8.2 The Central Insight

Our results crystallize into a single insight:

In-context learning can learn what attention can summarize

Attention mechanisms compute weighted averages over context positions. When a learning task's sufficient statistic is an aggregation over examples—a sum, mean, or weighted combination—attention computes it naturally, and ICL succeeds. When the task requires identifying

combinatorial structure through sequential search, attention's parallel averaging cannot help, and ICL fails regardless of scale.

This insight connects three disparate fields: learning theory (sufficient statistics), transformer architecture (attention as aggregation), and circuit complexity (AC^0 limitations). The synthesis reveals that ICL limitations are not merely empirical observations but mathematical necessities.

8.3 Implications and Perspective

Our framework provides practitioners a diagnostic tool (analyze sufficient statistics before deploying ICL), architecture designers specific guidance (depth for sequential tasks, hybrid architectures for hard problems), and theorists a unified language connecting learning theory, transformer expressivity, and circuit complexity. The dichotomy between ICL-Easy and ICL-Hard reflects a deeper truth: some problems admit parallel solutions while others are inherently sequential. Understanding these boundaries enables more effective deployment of transformer-based systems.

9. Acknowledgement

AI tools were employed as efficiency-enhancing assistants. The author declares no conflict of interest. This study received no external funding.

10. References

1. Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., & Zhou, D. (2023). What learning algorithm is in-context learning? Investigations with linear models. In *Proceedings of the 11th International Conference on Learning Representations*. https://openreview.net/forum?id=0g0X4H8yN4I
2. Allender, E., & Gore, V. (1993). A uniform circuit lower bound for the permanent. *SIAM Journal on Computing*, *23*(5), 1026–1049. https://doi.org/10.1137/S0097539792233907
3. Bai, Y., Chen, F., Wang, H., Xiong, C., & Mei, S. (2023). Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems* (Vol. 36, pp. 11014–11036). Curran Associates.
4. Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.
5. Bhattamishra, S., Ahuja, K., & Goyal, N. (2020). On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing* (pp. 7096–7116). Association for Computational Linguistics.
6. Blum, A., Kalai, A., & Wasserman, H. (2003). Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, *50*(4), 506–519.
7. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1877–1901). Curran Associates.

8. Chan, S. C., Santoro, A., Lampinen, A. K., Wang, J. X., Singh, A., Richemond, P. H., & Hill, F. (2022). Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 18878–18891). Curran Associates.
9. Coda-Forno, J., Binz, M., Akata, Z., Botvinick, M., Wang, J. X., & Schulz, E. (2023). Meta-in-context learning in large language models. In *Advances in Neural Information Processing Systems* (Vol. 36, pp. 60527–60541). Curran Associates.
10. Dziri, N., Lu, X., Sclar, M., Li, X. L., Jiang, L., Lin, B. Y., West, P., Bhagavatula, C., Le Bras, R., Hwang, J. D., Sanyal, S., Welleck, S., Ren, X., Ettinger, A., Harchaoui, Z., & Choi, Y. (2023). Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems* (Vol. 36, pp. 72610–72635). Curran Associates.
11. Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1126–1135). PMLR. https://proceedings.mlr.press/v70/finn17a.html
12. Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A*, *222*(594-604), 309–368. https://doi.org/10.1098/rsta.1922.0009
13. Furst, M., Saxe, J. B., & Sipser, M. (1984). Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, *17*(1), 13–27. https://doi.org/10.1007/BF01744431
14. Garg, S., Tsipras, D., Liang, P., & Valiant, G. (2022). What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 30583–30598). Curran Associates.
15. Hahn, M. (2020). Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, *8*, 156–171. https://doi.org/10.1162/tacl_a_00306
16. Halmos, P. R., & Savage, L. J. (1949). Application of the Radon-Nikodym theorem to the theory of sufficient statistics. *The Annals of Mathematical Statistics*, *20*(2), 225–241. https://doi.org/10.1214/aoms/1177730032
17. Håstad, J. (1987). *Computational limitations of small-depth circuits*. MIT Press.
18. Kearns, M. (1993). Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (pp. 392–401). https://doi.org/10.1145/167088.167200
19. Kharitonov, M. (1993). Cryptographic hardness of distribution-specific learning. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (pp. 372–381).
20. Kirsch, L., Harrison, J., Sohl-Dickstein, J., & Metz, L. (2022). General-purpose in-context learning by meta-learning transformers. In *NeurIPS Workshop on Foundation Models for Decision Making*.
21. Li, Y., Ildiz, M. E., Papailiopoulos, D., & Oymak, S. (2023). Transformers as algorithms: Generalization and stability in in-context learning. In *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 19513–19533). PMLR.
22. Maass, W., Schnitger, G., & Sontag, E. D. (1994). A comparison of the computational power of sigmoid and Boolean threshold circuits. In *Theoretical Advances in Neural Computation and Learning* (pp. 127–151).
23. Merrill, W., Sabharwal, A., & Smith, N. A. (2022). Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, *10*,

843–856. https://doi.org/10.1162/tacl_a_00493
24. Min, S., Lyu, X., Holtzman, A., Arber, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 11048–11064). Association for Computational Linguistics.
25. Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Yuan, Y., Hernandez, D., Chen, L., Kernion, A., Gadre, S., Kadavath, S., Jones, A., Johnston, J., Lasenby, B., Jackson, T., Dawson, E., . . . Olah, C. (2022). In-context learning and induction heads. *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html
26. Pérez, J., Barceló, P., & Marinkovic, J. (2021). Attention is Turing-complete. *Journal of Machine Learning Research*, *22*(75), 1–35. https://jmlr.org/papers/v22/20-302.html
27. Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, *56*(6), 1–40. https://doi.org/10.1145/1568318.1568324
28. Schmidhuber, J. (1987). *Evolutionary principles in self-referential learning* [Diploma thesis, Technische Universität München]. http://people.idsia.ch/~juergen/diploma.html
29. Thrun, S., & Pratt, L. (Eds.). (1998). *Learning to learn*. Springer.
30. Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142. https://doi.org/10.1145/1968.1972
31. Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, *16*(2), 264–280. https://doi.org/10.1137/1116025
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 5998–6008). Curran Associates.
33. Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., & Vladymyrov, M. (2023). Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 35151–35174). PMLR. https://proceedings.mlr.press/v202/von-oswald23a.html
34. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., & Fedus, W. (2022). Emergent abilities of large language models. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=yzk3vmUHSv
35. Xie, S. M., Raghunathan, A., Liang, P., & Ma, T. (2022). An explanation of in-context learning as implicit Bayesian inference. In *Proceedings of the 10th International Conference on Learning Representations*. https://openreview.net/forum?id=m7U_BqM6S7H
36. Yao, S., Peng, B., Papadimitriou, C., & Narasimhan, K. (2021). Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics* (pp. 3770–3785). Association for Computational Linguistics.
37. Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., & Kumar, S. (2020). Are transformers universal approximators of sequence-to-sequence functions? In *Proceedings of the 8th International Conference on Learning Representations*. https://openreview.net/forum?id=Bygj81BtPB

11. Appendices

Appendix A: FFN Approximation Lemmas

This appendix provides the technical lemmas establishing that feedforward networks can approximate the operations required for our transformer constructions.

Lemma A.1 (Monomial Computation). A ReLU network with depth O(log k) and width O(k) can compute the monomial x^k to accuracy ε on [0, 1], using O(k log(1/ε)) parameters.

Proof. We use the identity x^k = x^{k/2} · x^{k/2} for even k (and x · x^{k-1} for odd k). This gives a binary recursion of depth log k. Each multiplication x · y on [0, 1] can be approximated by a ReLU network: the identity xy = (1/4)[(x+y)² - (x-y)²] reduces multiplication to squaring, and x² on [0, 1] can be approximated by a width-O(1/ε) network using piecewise linear approximation. Composing log k multiplications with each having error ε/k gives total error O(ε) by sub-multiplicativity. Total depth: O(log k) layers, width O(k/ε) per layer, parameters O(k log(1/ε)). ■

Lemma A.2 (Matrix Multiplication). A ReLU network with depth O(log d) and width O(d³) can compute the product of two d × d matrices A, B with entries in [-B, B] to accuracy ε, using O(d³ log(dB/ε)) parameters.

Proof. Matrix multiplication (AB)_{ij} = Σ_k A_{ik} B_{kj} requires d³ products and d² sums of d terms each. Each product A_{ik} B_{kj} uses the multiplication construction from Lemma A.1 (constant depth for bounded entries). The d products per output entry are summed using a binary tree of depth log d. Width: O(d³) to compute all products in parallel. Depth: O(1) for products + O(log d) for summation = O(log d). Precision: each multiplication has error ε/(d²·d) = ε/d³; summing d terms gives error dε/d³ = ε/d². Total d² entries gives cumulative error O(ε). ■

Lemma A.3 (Matrix Inversion via Newton). For a symmetric positive definite matrix G with condition number κ, a ReLU network with depth O((log κ + log log(κ/ε)) · log d) and width O(d³) can compute G^{-1} to accuracy ε (in spectral norm).

Proof. The Newton-Schulz iteration Z_{t+1} = Z_t(2I - GZ_t) converges to G^{-1} in t = O(log κ + log log(κ/ε)) iterations (Lemma 3.5 in main text). Each iteration requires: (1) matrix multiplication GZ_t (Lemma A.2: depth O(log d)); (2) subtraction 2I - GZ_t (depth O(1)); (3) matrix multiplication Z_t(·) (Lemma A.2: depth O(log d)). Total per iteration: O(log d). Total for t iterations: O(t · log d) = O(log κ · log(κ/ε) · log d). Width O(d³) throughout. Precision analysis: Newton iteration is self-correcting, so numerical errors do not accumulate exponentially. With O(log(d/ε)) bits per entry, total error remains O(ε). ■

Lemma A.4 (Universal Approximation). For any continuous function g : [0,1]^k → ℝ and ε > 0, there exists a ReLU network with depth O(k) and width O((1/ε)^k) that approximates g uniformly to accuracy ε.

Proof. By the Weierstrass approximation theorem, g can be uniformly approximated by a polynomial of degree O(1/ε^{1/k}). This polynomial has O((1/ε)^k) monomials. Each monomial

is computed by Lemma A.1, and results are summed. The construction achieves depth O(k · log(1/ε)) and width O((1/ε)^k). For Lipschitz continuous g with constant L, width O((L/ε)^k) suffices. ■

Appendix B: Precision Analysis

This appendix provides detailed precision analysis for transformer operations.

Lemma B.1 (Stable Softmax). For scores s_1, ..., s_n with max_i |s_i| ≤ S, the softmax α_i = exp(s_i) / Σ_j exp(s_j) can be computed to accuracy ε using O(log n + S + log(1/ε)) bits of precision.

Proof. Use the numerically stable formula: α_i = exp(s_i - s_max) / Σ_j exp(s_j - s_max) where s_max = max_j s_j. After shifting, arguments to exp lie in [-2S, 0]. Values exp(s_i - s_max) lie in [exp(-2S), 1]. Computing exp to relative accuracy ε requires O(log(1/ε)) bits. The sum of n terms requires O(log n) additional bits to avoid overflow. Division requires O(log(1/ε)) bits for relative accuracy. Total: O(log n + S + log(1/ε)) bits. For S = O(log n) (Assumption A4), this is O(log n + log(1/ε)). ■

Lemma B.2 (Precision for Softmax Attention). Under Assumptions A1-A4, attention output o_i = Σ_j α_{ij} v_j can be computed to accuracy ε with p = O(log(n·d·B_v/ε)) bits, where B_v bounds value norms. B_v = ‖W_V‖ · B_x bounds value vector norms (following from Assumptions A2-A3).

Proof. By Lemma B.1, softmax weights α_{ij} are computed to accuracy $\varepsilon_1 = \varepsilon/(n \cdot B_v)$ with O(log n + log(n·B_v/ε)) bits. The weighted sum Σ_j α_{ij} v_j involves n terms, each bounded by B_v. Errors accumulate additively: error ≤ $n \cdot \varepsilon_1 \cdot B_v = \varepsilon$. Total precision: O(log(n·d·B_v/ε)). Under our assumptions, B_v = poly(n, d), so p = O(log n + log d + log(1/ε)) = O(log n) for constant ε. ■

Lemma B.3 (Score Bound Verification). Under Assumptions A2 (‖x‖ ≤ B_x) and A3 (‖W‖ ≤ B_W), attention scores satisfy |s_{ij}| = |q_i^T k_j / √d_k| ≤ $B_W^2 B_x^2 / \sqrt{d_k} = O(\text{poly}(n)/\sqrt{d})$.

Proof. Query: q_i = W_Q x_i, so ‖q_i‖ ≤ ‖W_Q‖ · ‖x_i‖ ≤ B_W · B_x. Similarly, ‖k_j‖ ≤ B_W · B_x. Inner product: |q_i^T k_j| ≤ ‖q_i‖ · ‖k_j‖ ≤ $B_W^2 B_x^2$. Scaling: |s_{ij}| ≤ $B_W^2 B_x^2 / \sqrt{d_k}$. With B_W, B_x = poly(n) and d_k = Θ(d), we get |s_{ij}| = O(poly(n)/√d). Standard initialization gives E[s_{ij}] = 0 and Var[s_{ij}] = O(1), so typical scores are O(1). ■

Corollary B.4 (AC^0 Containment). With p = O(log n) bits of precision and Assumptions A1-A4, all intermediate computations in a bounded-depth transformer can be performed in AC^0.

Proof. By Lemmas B.1-B.3, all quantities have polynomially bounded magnitudes and can be computed with O(log n) bits. By Lemmas 4.6-4.7 (main text), p-bit arithmetic is in AC^0 for p = O(log n). Composing O(1) layers of AC^0 operations yields AC^0 overall. ■

Appendix C: Complete Notation Reference

This appendix provides a comprehensive notation reference for the paper.

Table C.1: Complete Notation Reference

Symbol	Type/Domain	Description
X	Set	Input space
Y	Set	Output/label space
F	$F \subseteq \{f: X \to Y\}$	Function class under consideration
[n]	Set	{1, 2, ..., n}
C(n,k)	$\mathbb{N}$	Binomial coefficient n!/(k!(n-k)!)
n	$\mathbb{N}$	Number of in-context demonstration examples
d	$\mathbb{N}$	Input dimension / embedding dimension
k	$\mathbb{N}$	Sufficient statistic dimension / sparsity
ε	(0,1)	Accuracy parameter (error tolerance)
δ	(0,1)	Confidence parameter (failure probability)
L	$\mathbb{N}$	Transformer depth (number of layers)
H	$\mathbb{N}$	Number of attention heads
d_model	$\mathbb{N}$	Model/embedding dimension
d_ff	$\mathbb{N}$	Feedforward hidden dimension
d_k, d_v	$\mathbb{N}$	Key and value dimensions per head
p	$\mathbb{N}$	Numerical precision (bits)
s	$\mathbb{N}$	Total transformer size (parameters)
T(·)	$(X \times Y)^n \to \mathbb{R}^k$	Sufficient statistic function
φ(·,·)	$X \times Y \to \mathbb{R}^k$	Feature map for additive statistics
g(·,·)	$\mathbb{R}^k \times X \to Y$	Predictor function
L(·,·)	$Y \times Y \to \mathbb{R}_{\geq 0}$	Loss function
SSC	$\mathbb{N}$	Sufficient Statistic Complexity
AC-SSC	$\mathbb{N} \cup \{\infty\}$	Attention-Computable SSC
PC	$\mathbb{N}$	Prediction Complexity
n_ICL	$\mathbb{N}$	ICL sample complexity
n_ERM	$\mathbb{N}$	ERM sample complexity
G	$\mathbb{R}^{d \times d}$	Gram matrix X^TX
κ	$\mathbb{R}_{\geq 1}$	Condition number λ_max/λ_min
‖·‖	$\mathbb{R}_{\geq 0}$	Euclidean/spectral norm
‖·‖_F	$\mathbb{R}_{\geq 0}$	Frobenius norm
AC^0	Complexity class	Constant-depth poly-size circuits (AND/OR/NOT)
TC^0	Complexity class	AC^0 + threshold/majority gates
⊕	$\{0,1\}^k \to \{0,1\}$	XOR / parity function
O(·)	Function class	Asymptotic upper bound
Ω(·)	Function class	Asymptotic lower bound
Θ(·)	Function class	Asymptotically tight bound
poly(·)	Function class	Polynomial in argument